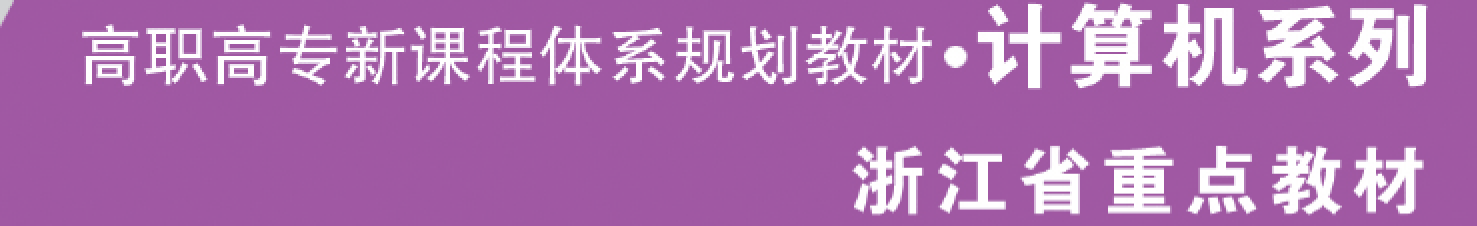

Web应用程序开发

程光华◎主编

徐　畅　谭恒松　张亚萍◎副主编

- 以任务为导向开展，适合项目化教学需要
- 贴近实际开发流程，注重实践理论的结合
- 涵盖ASP.NET的主要常用技术
- 覆盖Web开发中的典型功能模块
- 免费赠送教学PPT课件、课后习题答案

课件下载地址：www.tup.com.cn

清华大学出版社

高职高专新课程体系规划教材·计算机系列
浙江省重点教材

Web 应用程序开发

程光华 主 编

徐 畅 谭恒松 张亚萍 副主编

清华大学出版社

北 京

内 容 简 介

本教材采用项目化、任务驱动的模式编写，以一个电子商务平台的建设为案例，从平台的设计、开发、测试到部署，系统地讲解了使用 ASP.NET 进行 Web 应用程序开发的一般步骤和常用技术。

教材的主要内容包括基于 ASP.NET 的开发系统环境搭建、Web 系统开发流程、三层架构的搭建、页面框架的搭建、页面数据的显示、页面数据的更新、查询的实现、常用页面控件的使用、常用 Web 系统功能的实现、系统的配置和部署以及基于 Web 的系统测试等。

本教材以实际应用为出发点，归纳并选取了 Web 应用程序开发中具有代表性的知识和技能逐层深入进行讲解。本书适用于三年制高职高专、成人高校以及社会计算机培训学校等计算机及相关专业的学生使用，也可作为 Web 应用程序开发等人员的参考用书。

图书在版编目（CIP）数据

Web 应用程序开发/程光华主编. —北京：清华大学出版社，2011.12
高职高专新课程体系规划教材 • 计算机系列

ISBN 978-7-302-27538-1

I. ①W…　II. ①程…　III. ①互联网络-程序设计-高等职业教育-教材　IV. ①TP393.4

中国版本图书馆 CIP 数据核字（2011）第 254849 号

责任编辑：朱英彪
封面设计：刘　超
版式设计：文森时代
责任校对：张彩凤
责任印制：
出版发行：清华大学出版社　　**地　　址：**北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编：**100084
社 总 机：010-62770175　　**邮　　购：**010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：185×260　**印　　张：**12.75　**字　　数：**292 千字
版　　次：2011 年 12 月第 1 版　**印　　次：**2011 年 12 月第 1 次印刷
印　　数：1～4000
定　　价：28.00 元

产品编号：041951-01

前　　言

本教材从实际应用出发，结合多年基于 ASP.NET 的 Web 应用程序开发课程的教学经验编写完成。在内容设置上，归纳并总结了实际开发中采用的具有代表性的知识技能，并通过一个完整的 Web 系统的实施将它们贯穿起来。在内容的选取上遵循“理论够用，实践为重”的原则，重点突出应用技能的训练。

教材采用项目化、任务驱动的模式编写，以一个电子商务平台的建设为案例，从平台的设计、开发、测试到部署，系统地讲解了使用 ASP.NET 进行 Web 应用程序开发的一般步骤和常用技术。每一章都以 Web 系统中某些特定功能的实现为目标，通过若干个任务的实施和完成来实现目标。每个任务都设有如下 5 个模块。

（1）任务分析。介绍任务中要解决的具体问题，以及通过何种手段来解决问题。

（2）相关知识。介绍完成该任务需要使用到的知识以及技能。

（3）任务实施。用引导的方式，通过采用相关知识中的内容来完成任务中需要解决的问题。

（4）任务评价。对任务完成情况进行自我总结和评价。

（5）任务拓展。留给读者的在该任务基础上进行的相关实践内容。

本教材共分为 11 章，总体结构如下。

第 1 章，ASP.NET 开发系统环境搭建。讲述了使用 ASP.NET 进行 Web 应用程序开发时需要搭建的环境，包括开发时需要使用的软件的安装和配置。

第 2 章，Web 系统开发流程。通过项目案例，对 Web 系统开发流程中几个主要阶段以及在每个阶段中需要完成的内容进行阐述。

第 3 章，搭建系统框架。讲解了 Web 系统开发中三层架构的设计方法，以及使用三层架构来搭建项目案例的系统框架。

第 4 章，搭建页面框架。讲解了通过使用母版、导航等工具来统一系统页面的结构和风格，以及实现页面导航等功能。

第 5 章，显示数据。介绍了 Web 应用程序开发中进行数据展示可采用的几种形式，以及相关控件的使用方法。

第 6 章，添加和更新数据。讲解了 Web 应用程序开发中添加和修改数据的方法。

第 7 章，查询数据。讲解了数据查询的实现方法，以及使用显示控件显示精确内容的方法。

第 8 章，完善页面功能。介绍了页面交互中常用功能的实现方法，包括页面验证码和富文本输入。

第 9 章，完善系统功能。介绍了 HttpHandler 以及 Web Service 的开发，并通过它们实

现数字水印以及查询服务。

第 10 章，系统配置和部署。讲解了使用站点管理等工具进行 Web 系统的配置和部署。

第 11 章，系统测试。介绍了针对 Web 系统的两种不同测试方法，以及如何使用这两种测试方法对 Web 系统进行测试。

本教材主要以全日制的高职高专类计算机及相关专业的学生为主要读者对象，同时也可以作为成人高校以及社会培训学校计算机及相关专业学生的教材用书，还可作为 Web 应用程序开发等人员的参考用书。

本教材是多人智慧的结晶，除了主编和副主编外，参与编写、整理的人员还有严良达、韦存存、葛茜倩、朱一玮和钮俊等。在此对本书的参编人员以及对本书的编写提供支持和帮助的各界人士表示感谢。

由于编者水平有限，书中难免有疏漏之处，敬请广大读者批评指正，以期不断改进。

编　者

目　　录

第 1 章　ASP.NET 开发系统环境搭建

技能目标

1．能安装 Microsoft Visual Studio 2005、Microsoft SQL Server 2005。

2．能安装和配置 IIS、配置基础开发环境。

3．能创建 ASP.NET 应用程序。

相关词汇

英 文 单 词	中 文 含 义	英 文 单 词	中 文 含 义
IDE		Enterprise Edition	
MSDN		Developer Edition	
partial		Standard Edition	
Express Edition		Workgroup Edition	

工作任务

任务 1.1　搭建系统开发环境

任务 1.1　搭建系统开发环境

1.1.1　任务分析

任务目标：将 Microsoft Visual Studio 2005、Microsoft SQL Server 2005 两个主要支持软件顺利地安装在带有 Microsoft Windows Server 2003 或者 Microsoft Windows XP 操作系统的 PC 开发机上；能够配置 Microsoft Visual Studio 2005 开发环境，并且能够创建一个 ASP.NET 应用程序。

完成标准：能够在目标 PC 开发机上安装好所需的软件，配置好环境，并能够创建 ASP.NET 应用程序。

应用手段：按需要安装支持 ASP.NET 技术的主要软件。

1.1.2　相关知识

1.1.2.1　ASP.NET 简介

1．什么是 ASP.NET

ASP.NET 是统一的 Web 应用程序平台，它提供了为建立和部署企业级 Web 应用程序所必需的服务。ASP.NET 为能够面向任何浏览器或设备的更安全、更强的可升级性及更稳定的应用程序，提供了新的编程模型和基础结构。

ASP.NET 是 Microsoft .NET Framework 的一部分，是一种可以在高度分布的 Internet 环境中简化应用程序开发的计算环境。.NET Framework 包含公共语言运行库，它提供了各种核心服务，例如，内存管理、线程管理和代码安全；它还包含.NET Framework 类库，这是一个开发人员用于创建应用程序的综合的、面向对象的类型集合。

2．ASP.NET 的特色与优势

ASP.NET 具有以下特色与优势：

- 可管理性：ASP.NET 使用基于文本的、分级的配置系统，简化了将设置应用于服务器环境和 Web 应用程序的工作。因为配置信息是存储为纯文本的，因此可以在没有本地管理工具的帮助下应用新的设置。配置文件的任何变化都可以自动检测到并应用于应用程序。
- 安全：ASP.NET 为 Web 应用程序提供了默认的授权和身份验证方案。开发人员可以根据应用程序的需要进行添加、删除或替换这些方案。
- 易于部署：通过将必要的文件复制到服务器上，ASP.NET 应用程序即可以部署到该服务器上。不需要重新启动服务器，甚至在部署或替换运行的已编译代码时也不需要重新启动。
- 增强的性能：ASP.NET 是运行在服务器上的已编译代码。与传统的 Active Server Pages (ASP) 不同，ASP.NET 能利用早期绑定、实时（JIT）编译、本机优化和全新的缓存服务来提高性能，编译的过程如图 1-1 所示。

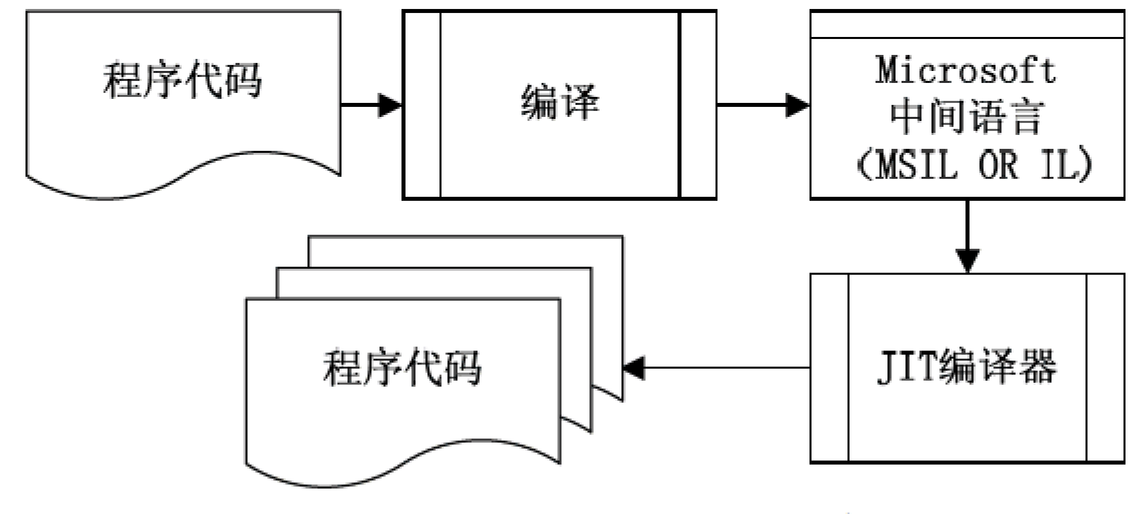

图 1-1　ASP.NET 页面编译

- 灵活的输出缓存：根据应用程序的需要，ASP.NET 可以缓存页数据、页的一部分或整个页。缓存的项目可以依赖于缓存中的文件或其他项目，或者可以根据过期策略进行刷新。
- 移动设备支持：ASP.NET 支持任何设备上的任何浏览器。开发人员使用与用于传

统的桌面浏览器相同的编程技术来处理新的移动设备。

- 扩展性和可用性：ASP.NET 被设计成可扩展的、具有特别专有的功能来提高群集的、多处理器环境的性能。此外，Internet 信息服务（IIS）和 ASP.NET 运行时密切监视和管理进程，以便在一个进程出现异常时，可在该位置创建新的进程，以便应用程序继续处理请求。
- 跟踪和调试：ASP.NET 提供了跟踪服务，该服务可在应用程序级别和页面级别调试过程中启用。可以选择查看页面的信息，或者使用应用程序级别的跟踪查看工具查看信息。在开发和应用程序处于生产状态时，ASP.NET 支持使用.NET Framework 调试工具进行本地和远程调试。当应用程序处于生产状态时，跟踪语句能够留在产品代码中而不会影响性能。
- 与.NET Framework 集成：因为 ASP.NET 是.NET Framework 的一部分，整个平台的功能和灵活性对 Web 应用程序都是可用的。开发人员也可以从 Web 上访问.NET 类库、查看消息和数据访问解决方案。ASP.NET 是独立于语言之外的，所以开发人员能选择最适于应用程序的语言。
- 与现有 ASP 应用程序的兼容性：ASP 和 ASP.NET 可并行运行在 IIS Web 服务器上而互不冲突；不会发生因安装 ASP.NET 而导致现有 ASP 应用程序崩溃的情况。ASP.NET 仅处理具有.aspx 文件扩展名的文件，具有.asp 文件扩展名的文件仍由 ASP 引擎来处理。然而，应该注意的是会话状态和应用程序状态并不在 ASP 和 ASP.NET 页面之间共享。

ASP.NET 启用了分布式应用程序的两个功能，即 Web 窗体和 XML Web 服务，相同的配置和调试基本结构都支持这两种功能。Web 窗体技术帮助用户建立强大的基于窗体的网页，Web 窗体页面则使用可重复使用的内建组件或自定义组件来简化页面中的代码。

使用 ASP.NET 创建的 XML Web 服务，可以远程访问服务器。使用 XML Web 服务，商家可以提供其数据或商业规则的可编程接口，然后可以由客户端和服务器端的应用程序获得和操作。通过在客户端/服务器和服务器/服务器方案中的防火墙范围内的使用标准（如 XML 消息处理和 HTTP 等），XML Web 服务可启用数据交换。以任何语言编写的且运行在任何操作系统上的程序都能调用 XML Web 服务。

3．ASP.NET 的典型应用

微软网站（www.microsoft.com/en/us/default.aspx）是最早采用 ASP.NET 技术的网站，随着全球用户的增加，证明 ASP.NET 技术是能够满足现代市场需求的，也可以应对高标准、高要求的企业应用。

当当网（www.dangdang.com）是全球最大的中文网上书店，当当网的所有商品都是通过网上店铺进行销售的，这是 ASP.NET 技术在 B2C 的成功应用。

CSDN 网（www.csdn.com）是全球最大的中文计算机技术论坛网站，该网站的用户量大，论坛种类多，网站运行稳定，这也是 ASP.NET 技术的成功应用。

1.1.2.2　Visual Studio 2005 IDE

Visual Studio 2005 IDE（Integrated Develop Environment，集成开发环境）提供了比 Visual

Studio .NET 2003 IDE 更加方便、友好的开发 ASP.NET 应用程序的环境，具体有以下几点。

1．内置服务器，不再依赖 IIS

在 Visual Studio .NET 2003 集成开发环境中，开发 ASP.NET Web 应用程序必须依赖 IIS，而在 Visual Studio 2005 中，IIS 则成为可选项。Visual Studio 2005 自身提供了一个用于开发、调试 ASP.NET Web 应用程序的内置 Web 服务器。在 Visual Studio 2005 中直接运行.NET 应用程序时，该内置 Web 服务器就会自动启动，这时可以在 Windows 的任务栏中看到内置服务器的小图标，双击该图标，可看到如图 1-2 所示的内置 Web 服务器的详细情况。

图 1-2　内置 Web 服务器

2．访问网站方法多样

在 Visual Studio .NET 2003 中打开一个网站或 ASP.NET 应用程序，必须在 IIS 中配置该网站对应的虚拟目录，而在 Visual Studio 2005 中则是可选的。在 Visual Studio 2005 中，可以使用本地的文件系统、FTP 站点或 IIS 站点等方式来访问一个网站或 ASP.NET 应用程序。

3．简单的网站发布

在 Visual Studio 2005 中发布网站方法更简单，只需在应用程序开发的“解决方案资源管理器”面板中右击解决方案名称，并在弹出的快捷菜单中选择“发布网站”命令，弹出如图 1-3 所示的“发布网站”对话框，单击“确定”按钮，就可以很方便地完成网站发布操作。

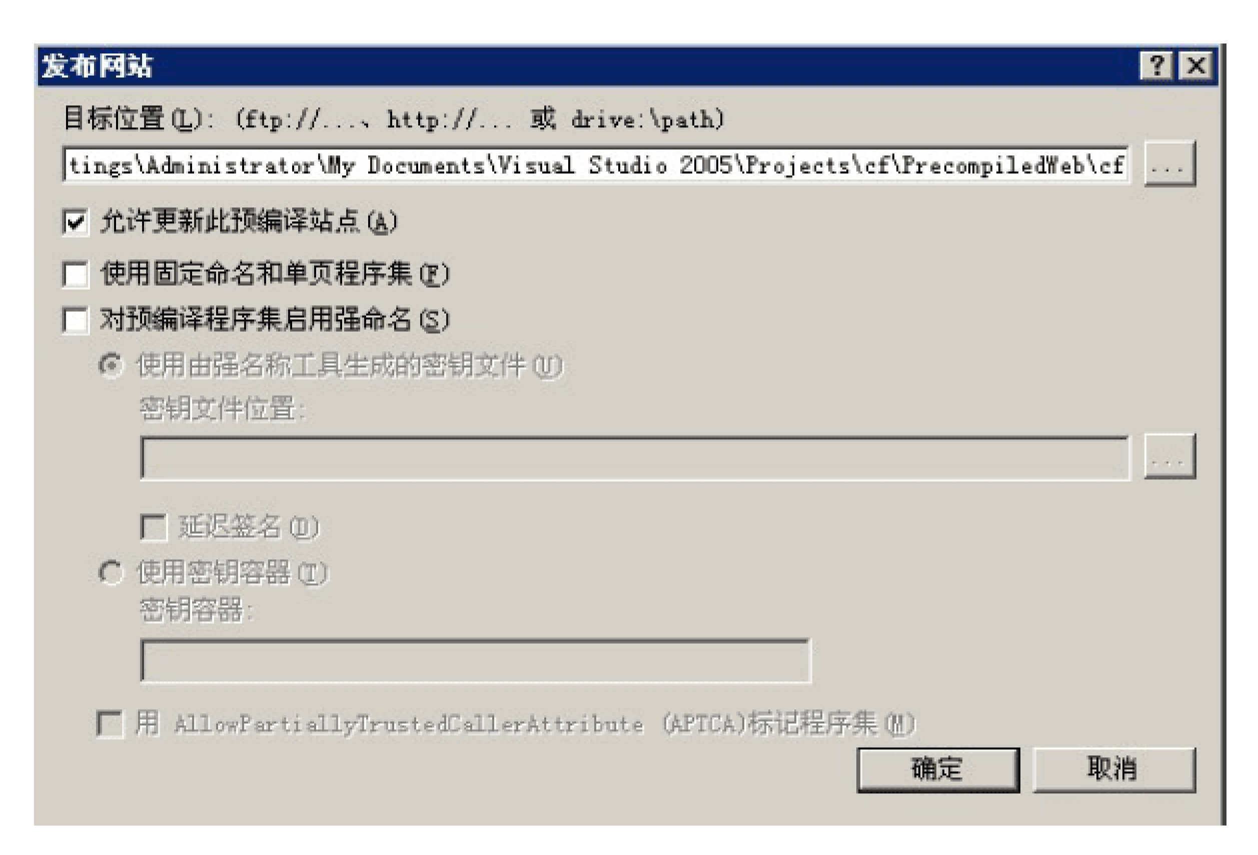

图 1-3　“发布网站”对话框

4．网站复制

Visual Studio 2005 提供了网站复制功能。使用该功能，可以方便、快捷地备份网站的资源（如代码、图像等）。单击“解决方案资源管理器”面板中的“复制网站”按钮，即可复制“解决方案资源管理器”面板中当前正在开发的网站。

1.1.3　任务实施

1．安装 Microsoft Visual Studio 2005

要在 Visual Studio 2005 IDE 环境下开发 ASP.NET Web 应用程序，必须要先安装该开发环境。下面介绍安装的具体步骤，以 Windows Server 2003 操作系统为例。

（1）双击 Visual Studio 2005 的安装应用程序 setup.exe，出现“Visual Studio 2005 安装程序”窗口，其中显示了“安装 Visual Studio 2005”、“更改或移除产品文档”和“检查 Service Release”3 个选项，这时，只有“安装 Visual Studio 2005”选项可用。

（2）选择“安装 Visual Studio 2005”选项，弹出 Microsoft Visual Studio 2005 安装向导窗口。此时，安装程序正在加载安装组件。

（3）单击“下一步”按钮，弹出 Microsoft Visual Studio 2005 安装程序—起始页窗口。

（4）单击“下一步”按钮，弹出“Microsoft Visual Studio 2005 安装程序—选项页”窗口，在此可以选择要安装的功能以及安装路径。这里选择“自定义”选项，并配置功能、设置安装路径，如图 1-4 所示。

图 1-4　自定义安装选择

（5）单击“安装”按钮，弹出“Microsoft Visual Studio 2005 安装程序—安装页”窗口，当前窗口显示安装程序的进度，如图 1-5 所示。

图 1-5　安装 Visual Studio 2005

（6）安装完成后，弹出“Microsoft Visual Studio 2005 安装程序—完成页”窗口。

（7）单击“完成”按钮，弹出提示安装完成的窗口。此时，“更改或移除产品文档”和“检查 Service Release”是可用选项，此时也可以选择安装 MSDN（Microsoft Developer Network）。

知识点小贴士

MSDN 是微软公司面向软件开发者的一种信息服务，它是一个以 Visual Studio 和 Windows 平台为核心整合的开发虚拟社区，包括技术文档、在线电子教程、网络虚拟实验室、微软产品下载（几乎包括全部的操作系统、服务器程序、应用程序和开发程序的正式版和测试版，还包括各种驱动程序开发包和软件开发包）、Blog、BBS、MSDN WebCast 以及与 CMP 合作的 MSDN 杂志等一系列服务。MSDN 的中文网址是 www.microsoft.com/china/msdn。

2．安装 Microsoft SQL Server 2005

Microsoft SQL Server 2005 包括以下 5 个版本，分别是：

- SQL Server 2005 Express Edition，免费版本，微软官方网站提供下载。
- SQL Server 2005 Enterprise Edition，企业版本。
- SQL Server 2005 Developer Edition，开发版本。
- SQL Server 2005 Standard Edition，标准版本。
- SQL Server 2005 Workgroup Edition，工作组版本。

Microsoft SQL Server 2005 的安装组件包括如下几种：

- SQL Server Database Engine，数据库引擎。
- Analysis Services，数据法分析服务。

- Reporting Services，报表服务。
- Notification Services，通知服务。
- Integration Services，集成服务。
- 管理工具、文档和实例。

其中：

SQL Server 2005 Enterprise Edition 旨在支持超大型企业的联机事务处理（OLTP）环境、高度复杂的数据分析需求、数据仓库系统和活跃的 Web 站点。若要设计大型数据库安装的 DBA，则只能考虑 Enterprise 版本。

SQL Server 2005 Standard Edition 包括大多数中小型企业使用的电子商务、数据仓库和业务线（line-of-business）应用所需的基础功能。对于不需要企业版全部功能的企业，可以考虑安装 Standard 版本。

SQL Server 2005 Workgroup Edition 适用于小型公司的数据管理解决方案，它们需要一个对数据量和用户数不加限制的数据库，并且能够充当小型 Web 服务器和部门或分支办公室操作的后端。在小型服务器上操作少量数据的 DBA 可以考虑使用 Workgroup 版本。

SQL Server 2005 Developer Edition 包括 SQL Server 2005 Enterprise 版本的全部功能，但它被授权用作一个开发和测试系统，而不是作为一个生产服务器。Developer 版本适合于大型公司中需要使用 Enterprise 版本开发应用程序，但是又不想在开发或测试服务器上安装 Enterprise 版本的开发人员。

SQL Server 2005 Express Edition 是一个免费、易用、易于管理的数据库，开发人员可以对它进行重新分布，以充当客户数据库以及基本的服务器数据库。Express 版本通常只适合于非常小的数据集。如果开发人员开发的应用程序需要一个小型数据存储库，那么应考虑使用 Express 版本。

本书将采用 SQL Server 2005 Standard Edition 来作为案例项目的后台数据库。

注意事项

在安装 Microsoft SQL Server 2005 时，如果是 Windows XP 操作系统，安装 Enterprise 版本时，只能安装管理工具、文档和示例，不能安装数据服务器，如 SQL Server Database Engine、Reporting Services 等。

本例是在 Windows Server 2003 操作系统环境下安装 Microsoft SQL Server 2005，具体安装步骤如下。

（1）双击 Microsoft SQL Server 2005 的安装程序 setup.exe，进入 SQL Server 2005 安装界面。单击“安装”栏中的“服务器组件、工具、联机丛书和示例”超链接，如图 1-6 所示。

（2）弹出“Microsoft SQL Server 2005 安装程序”对话框，选中“我接受许可条款和条件”复选框，单击“下一步”按钮，如图 1-7 所示。

图 1-6　SQL 安装向导

图 1-7　“安装程序”对话框

（3）在弹出的对话框中直接单击“安装”按钮，即可开始安装 SQL Server 组件，即 Microsoft SQL Native Client 和 Microsoft SQL Server 2005 安装程序支持文件，如图 1-8 所示。

（4）完成 SQL Server 组件安装之后，安装程序开始检测系统配置。

（5）检测系统配置完成之后，弹出“欢迎使用 Microsoft SQL Server 安装向导”对话框，如图 1-9 所示。

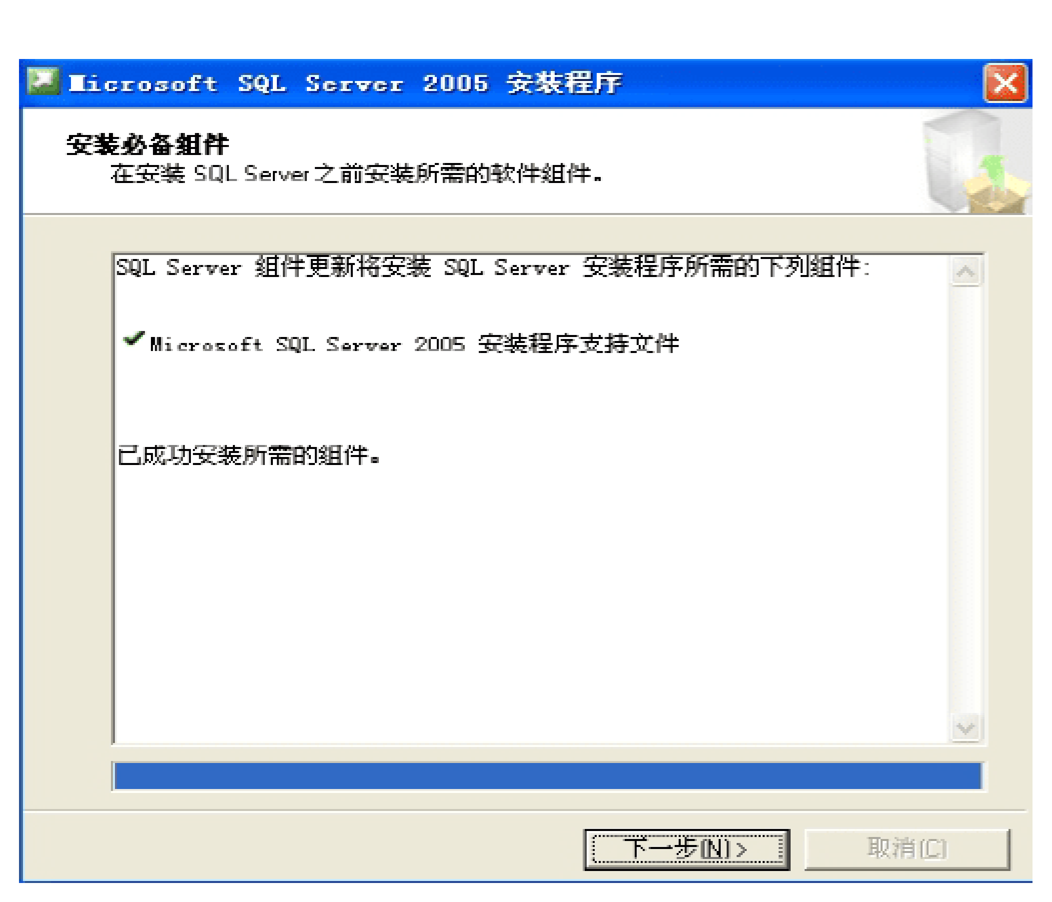

图 1-8　安装组件

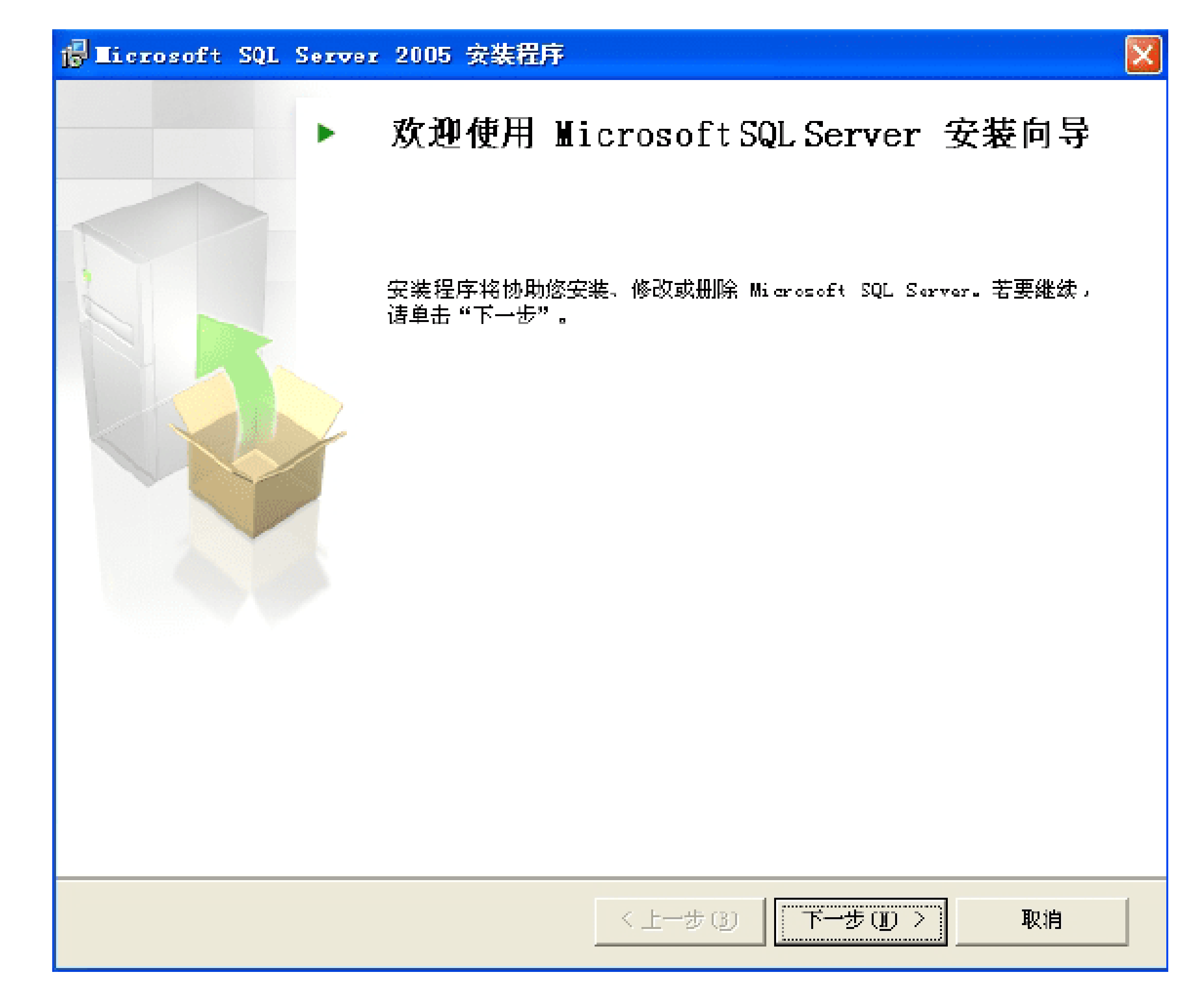

图 1-9　“安装向导”对话框

（6）单击“下一步”按钮，进入“系统配置检查”界面，在当前界面还显示了检测结果，如图 1-10 所示。若是正确安装，则应该有 14 项信息是成功状态。

（7）单击“下一步”按钮，安装程序处于准备安装状态。

（8）单击“下一步”按钮，在进入的界面中输入注册信息，包括姓名、公司和产品序列号等。

图 1-10　安装检查系统配置图

（9）单击“下一步”按钮，打开“要安装的组件”对话框，这里选择 SQL Server Database Services 数据库引擎组件、Reporting Services 报表服务组件和“工作站组件、联机丛书和开发工具”3 个组件，如图 1-11 所示。

（10）单击“下一步”按钮，打开“功能选择”对话框，在其中选择“文档、示例和示例数据库”下的所有功能，如图 1-12 所示。

图 1-11　安装组件选择

图 1-12　安装功能选择

（11）单击“下一步”按钮，打开“实例名”对话框，在其中选中“默认实例”单选按钮，如图 1-13 所示。

（12）单击“下一步”按钮，打开“服务账户”对话框，在其中选中“使用内置系统账户”单选按钮，并在其后的下拉列表框中选择“本地系统”选项，然后在“安装结束时启动服务”栏中选中 SQL Server 和 SQL Browser 两个复选框，如图 1-14 所示。

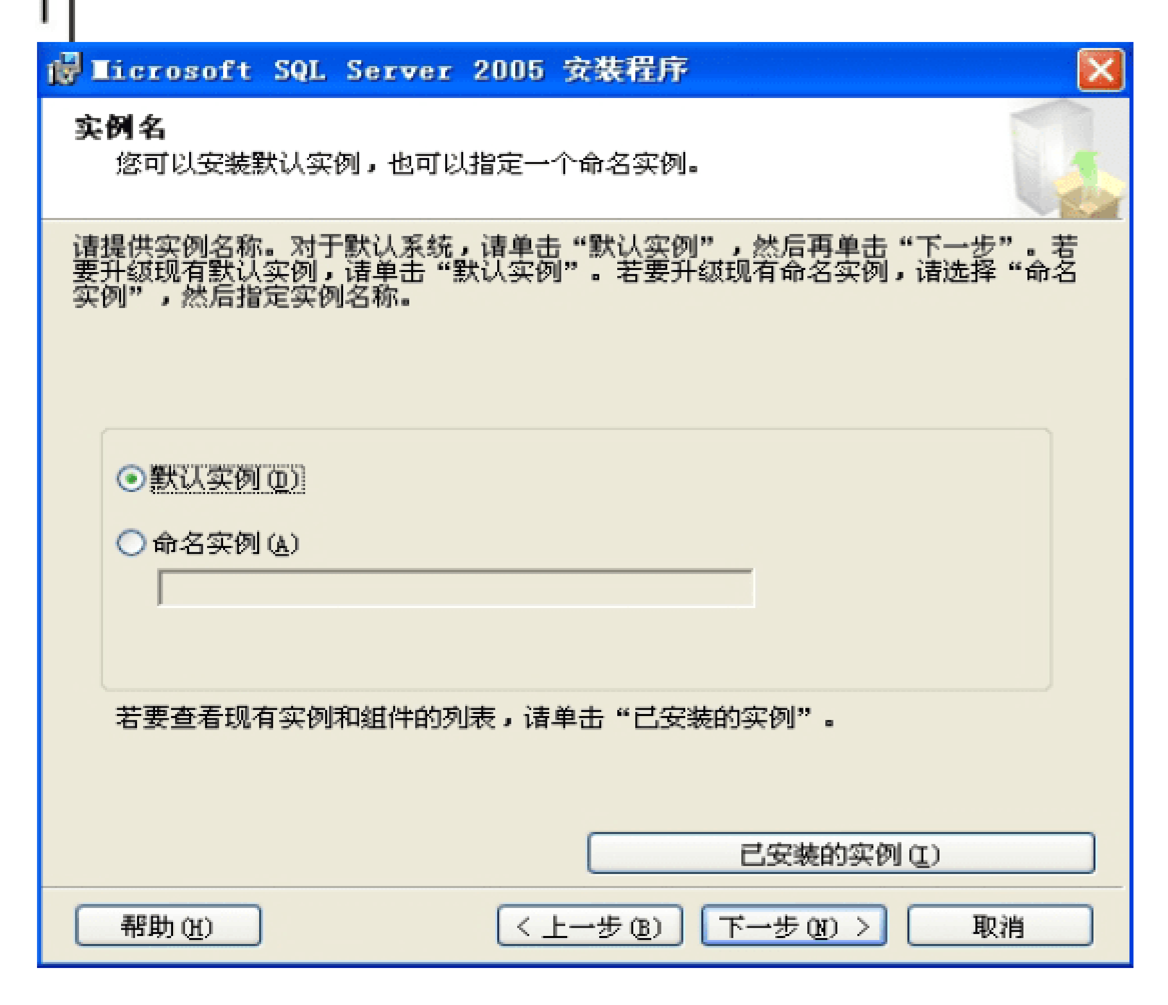

图 1-13　实例选择

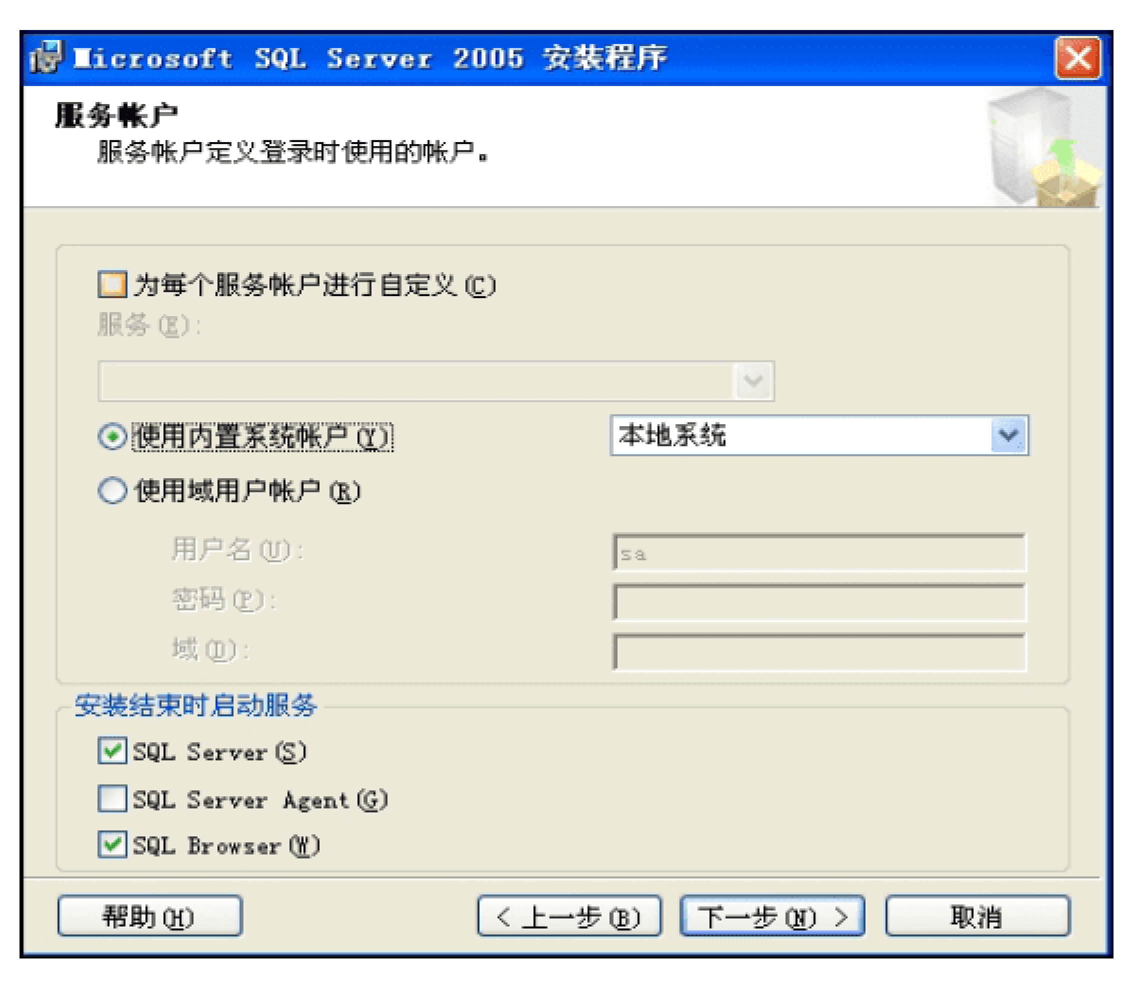

图 1-14　服务账户选择

（13）单击“下一步”按钮，打开“身份验证模式”对话框，在其中选中“混合模式（Windows 身份验证和 SQL Server 身份验证”单选按钮，并设置 sa 账户的密码，如图 1-15 所示。

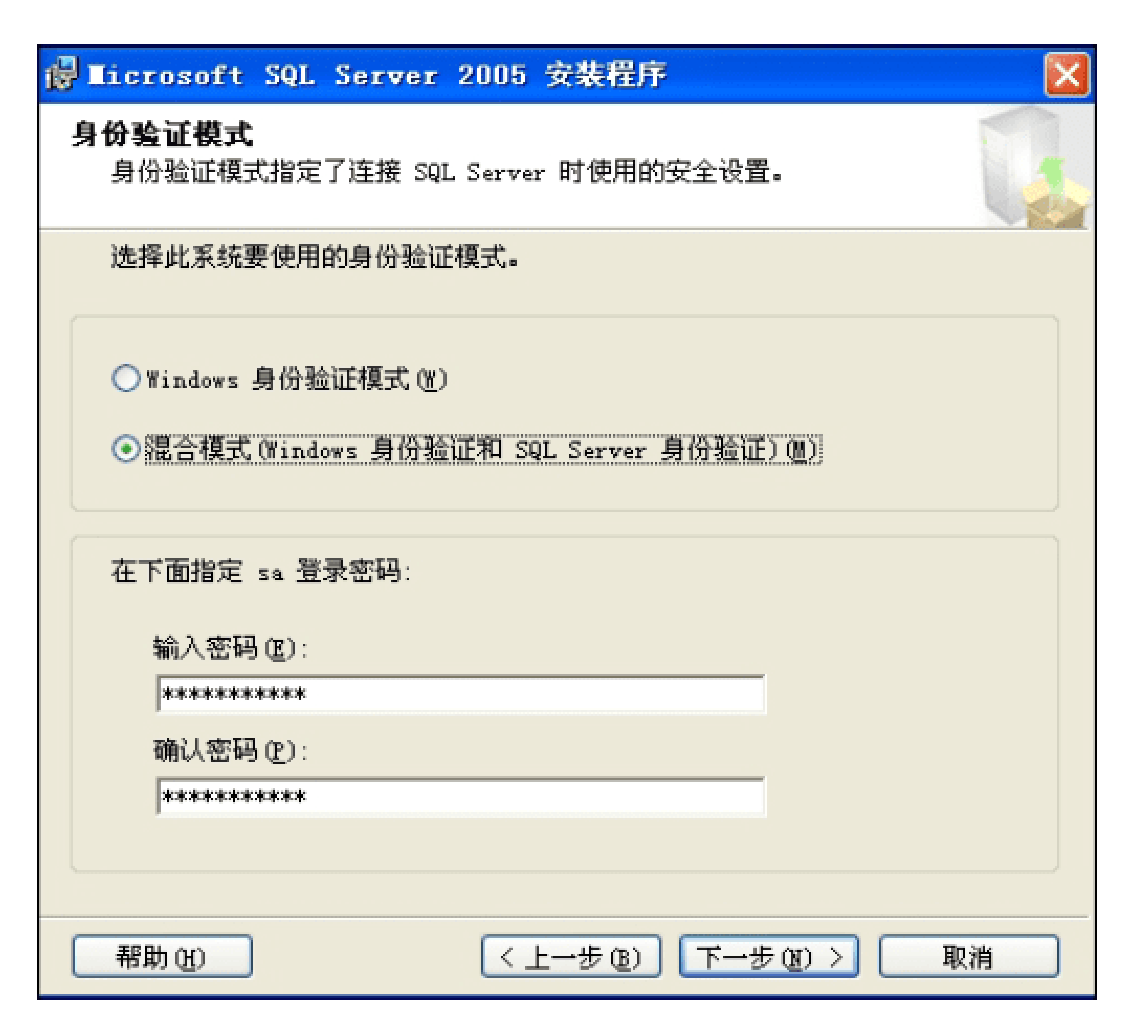

图 1-15　身份验证模式选择

（14）单击“下一步”按钮，在打开的对话框中配置排序规则，这里选择默认配置。

（15）单击“下一步”按钮，在打开的对话框中配置报表服务器安装选项，在其中选中“安装默认配置”单选按钮。

（16）单击“下一步”按钮，在打开的对话框中设置错误和使用情况报告选项，这里使用默认选择。

（17）单击“下一步”按钮，在打开的对话框中确认安装程序安装的组件。

（18）单击“安装”按钮，安装程序开始安装用户配置的组件。

（19）安装完成后，各个组件的状态都显示为“安装完毕”。

（20）单击“下一步”按钮，安装程序将完成整个安装过程，此时可以查看安装日

志等信息。安装成功后，可以在“开始”|“所有程序”菜单中查看 Microsoft SQL Server 2005。

3．IIS 的安装

IIS（Internet Information Services）是由 Microsoft 公司开发的 Web 服务器，它基于 Windows 操作系统，操作方便，功能强大，为 ASP.NET 提供稳定的运行环境。IIS 在操作系统安装时默认是不安装的组件，而且 IIS 根据操作系统的版本不同也有区别，所以要顺利安装 IIS，最好找到与当前操作系统一致的系统安装光盘。这里以 Windows XP 操作系统为例进行介绍。

注意事项

实际使用过程中，ASP.NET 程序发布，Web 服务一般是选择 Windows Server 操作系统的，不会选择 Windows XP 操作系统，Server 操作系统更加稳定，性能更加优越，功能更加全面。

IIS 的安装步骤如下：

（1）在 Windows 系统的“开始”菜单中选择并打开“控制面板”，双击“添加或删除程序”图标，在弹出的“添加或删除程序”窗口中选择“添加/删除 Windows 组件”选项卡。

（2）打开“Windows 组件向导”对话框，在其中选中“Internet 信息服务（IIS）”复选框，如图 1-16 所示（确保列表前的复选框被选中）。

（3）单击“详细信息”按钮，弹出“Internet 信息服务（IIS）”对话框，在其中选中“Internet 信息服务管理单元”和“万维网服务”选项，如图 1-17 所示。

图 1-16　选中“Internet 信息服务（IIS）”复选框

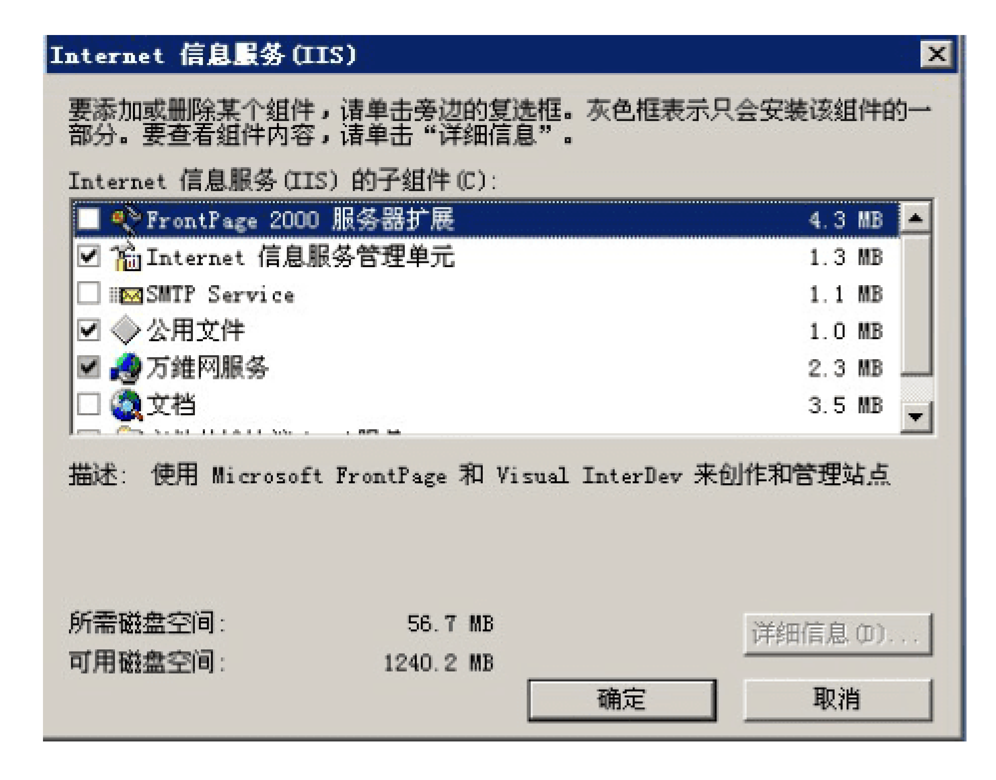

图 1-17　“Internet 信息服务（IIS）”对话框

（4）单击“确定”按钮，然后单击“下一步”按钮，系统将自动查找光盘，并配置组件，如图 1-18 所示。

（5）安装完成后，弹出“完成 Windows 组件向导”对话框，如图 1-19 所示。

4．配置集成开发环境

安装好 Microsoft Visual Studio 2005 之后，就可以使用 IDE（集成开发环境）了，但是

在正式开发 Web 应用程序之前，首次进入 IDE 时还要对 IDE 进行配置，也就是通过一些通用的设置，来提高用户的开发效率以便使程序资源的管理更加方便等。

图 1-18　配置组件

图 1-19　安装完成

（1）窗口布局。Microsoft Visual Studio 2005 IDE 中有许多窗口是可以悬停、浮动、停靠的，可以根据自己的习惯来设置窗口的布局，如图 1-20 所示。

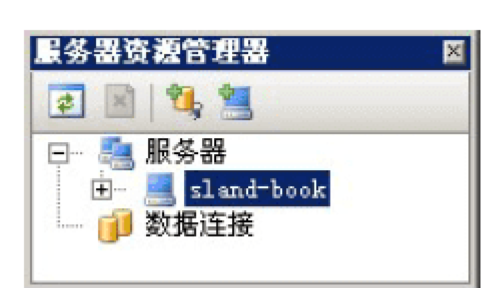

图 1-20　多种悬停浮动窗口

（2）常用选项。打开 Microsoft Visual Studio 2005 IDE，在菜单栏中选择“工具”|“选项”命令，打开如图 1-21 所示的“选项”对话框，在其中可以根据自己的习惯、开发机器等进行各项配置。例如，选择“文本编辑器”结点下的“C#”子结点，可以配置 C#编辑器的语句结束的属性、设置的属性和显示的属性等。选择“HTML 设计器”结点，可以配置 HTML 编辑的起始页位置的属性和智能标记的属性等。

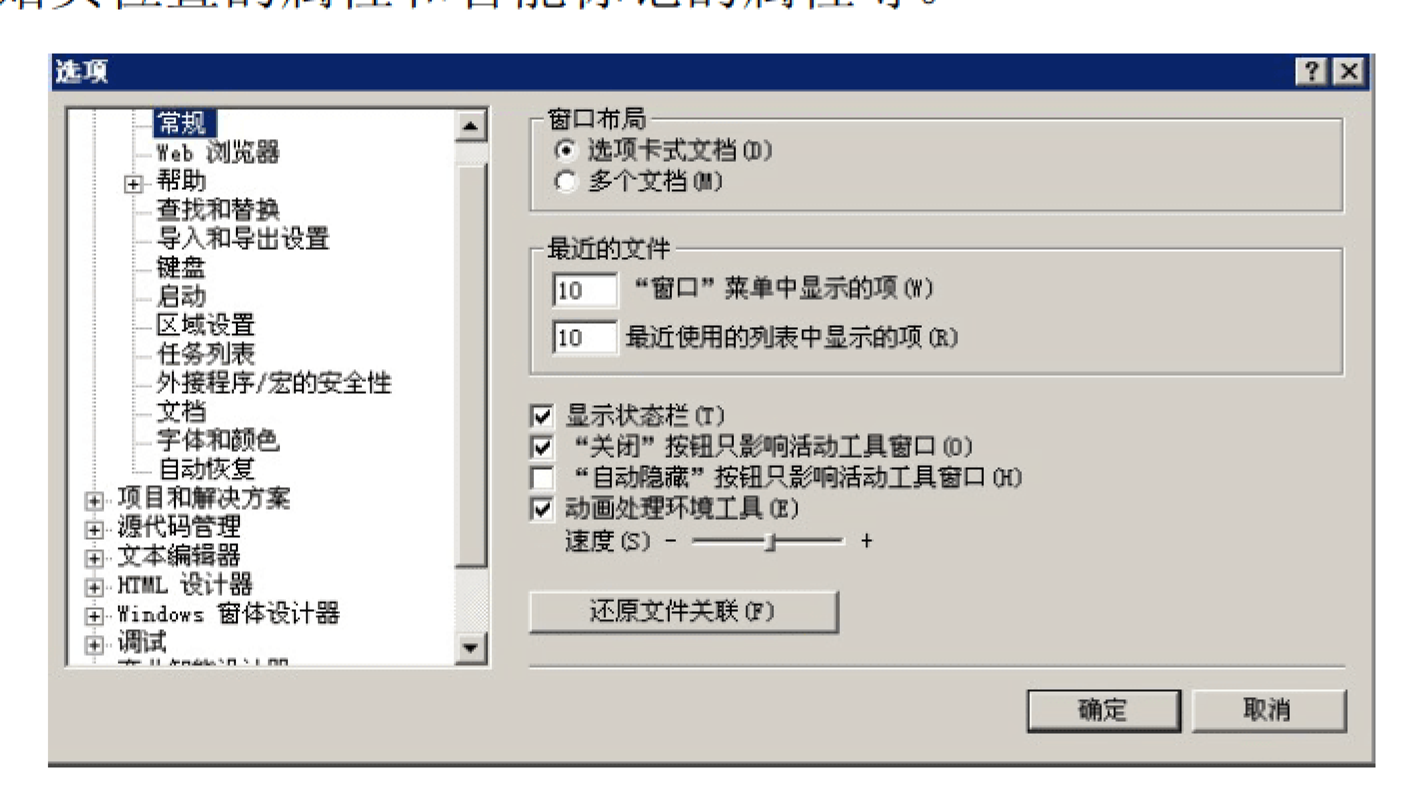

图 1-21　Microsoft Visual Studio 2005 IDE 选项配置

（3）设置的导入与导出。为了提供更加方便的 IDE 的配置功能，IDE 还提供了导入和导出开发环境设置的功能。在 Microsoft Visual Studio 2005 IDE 的菜单中，选择“工具”|

“导入和导出设置”命令，在打开的对话框中可以进行导入与导出环境的设置，通过它开发人员就可以很快捷地将开发环境设置为自己习惯的形式。

5．创建一个 ASP.NET 应用程序

（1）创建 ASP.NET 应用程序。打开 Visual Studio 2005，在菜单栏中选择“文件”|“新建”|“网站”命令，在打开的对话框中可以看到位置中并不需要指定网站路径，直接指定文件路径即可。在“位置”栏可以指定 3 种类型，通常使用文件系统，单击“浏览”按钮可以找到文件夹所在的位置，避免手动输入路径产生错误。同时，还需要在“语言”栏中指定开发的语言，如图 1-22 所示。设置好后单击“确定”按钮，即可完成一个网站的创建。

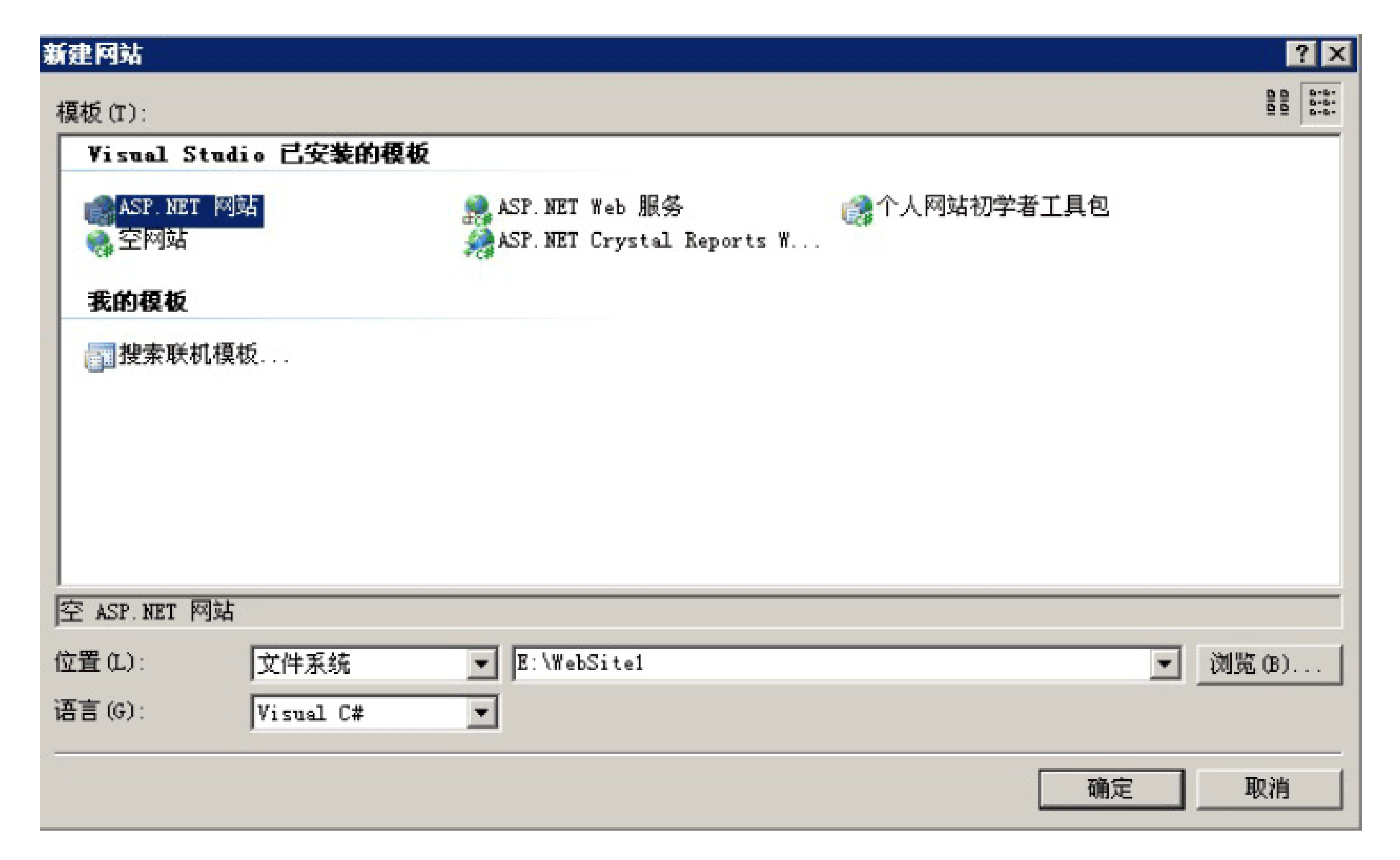

图 1-22　创建 ASP.NET 应用程序

事实上，Visual Studio 提供了 4 种 Web 站点位置的存放方式，即文件系统、本地 IIS、FTP 站点和远程站点。如果在图 1-22 中单击“浏览”按钮，即可看到如图 1-23 所示的“选择位置”对话框。在该对话框的左侧可以看到有 4 个不同的选项卡，其中：

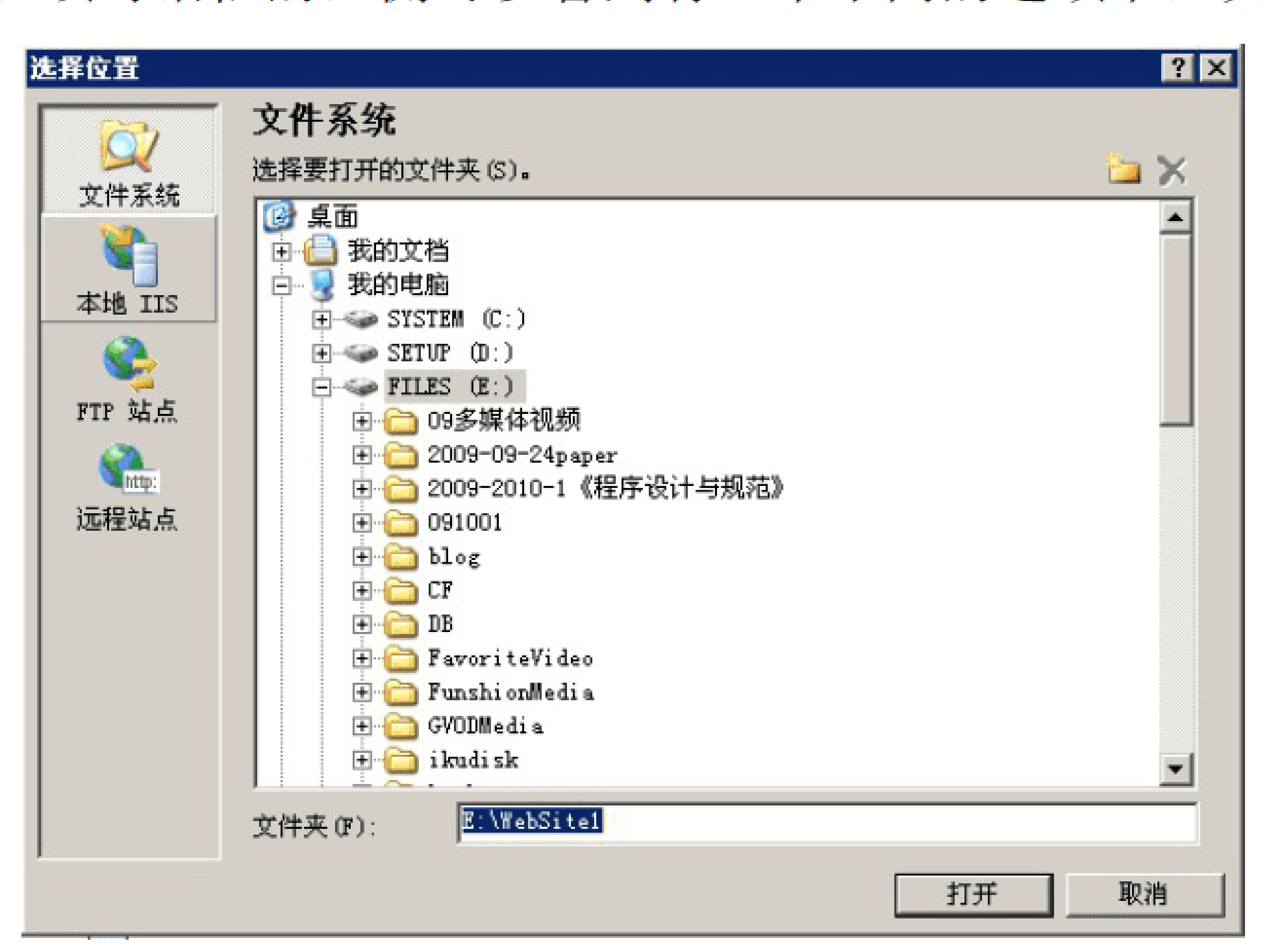

图 1-23　“选择位置”对话框

● 文件系统：允许将站点文件存储在本地硬盘的一个指定文件夹中，或者存在局域

网中的一个共享位置。这就意味着无须将站点作为 IIS 应用程序来创建，就可以进行开发与测试等工作。如果用户的计算机没有安装 IIS，又要开发 Web 服务程序，这种方式就非常合适。该方式的缺点是无法使用 IIS 的某些功能，毕竟开发的服务器是简化的 Web 服务器，一般在开发过程中使用这种方式。

- 本地 IIS：是早期的一种创建方式，比文件系统的方法复杂，但可以使用 IIS 的高级功能，所以常常在做 Web 测试时使用。缺点是 IIS 服务只有一个进程，调试时也只能是一个用户参与调试。
- FTP 站点：可以在服务器上保存文件，通过 FTP 访问它们。这是共享环境中的一种可能配置，在共享环境下许多人可以同时使用项目。另外，还可以使用 FTP 设置远程编辑文件。其中，远程服务器 IIS 有一个虚拟目录映射到 FTP 文件驻留位置。该方式的缺点是不能使用源代码管理器（如 VSS- Visual Source Safe 微软公司的版本控制管理软件）。
- 远程站点：与 FTP 站点类似，所不同的是不再使用 FTP 方式连接，而是采用 HTTP 方式。这种方式配置站点相当复杂，而且缺点比较多，一般很少使用。

站点创建好后，IDE 自动添加了一个 Default.aspx 页面，它有两种编辑方式：设计视图和源视图。单击“运行”按钮（或者按 F5 键），即可运行程序，因为没有编写任何代码，所以运行之后是一个空白的页面。

注意事项

第一次运行新建的站点时，会弹出如图 1-24 所示的“未启用调试”对话框，此时没有启动调试，但是可以自动添加调试，单击“确定”按钮，就自动启动了调试功能。

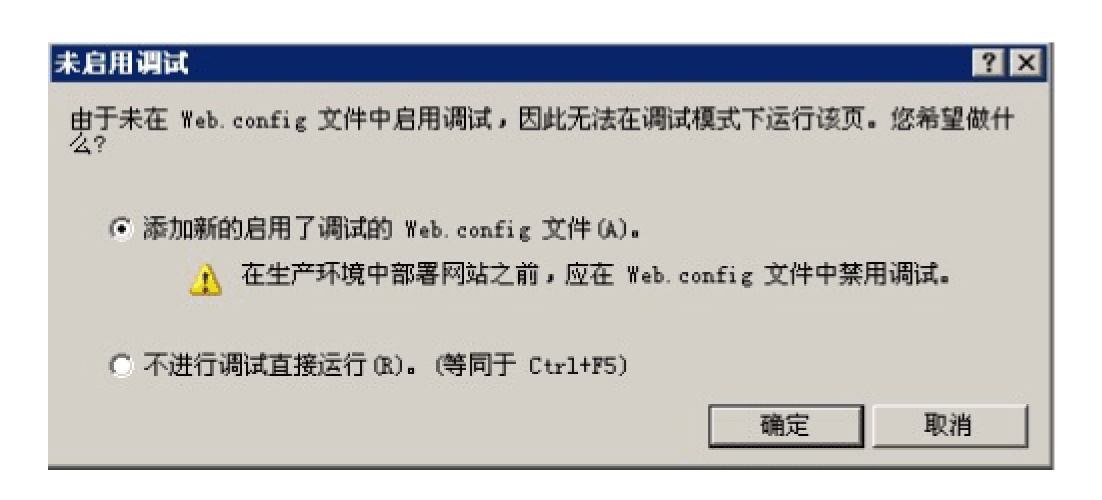

图 1-24 “未启用调试”对话框

知识点小贴士

Web.config 文件是一个 XML 文本文件，它用来储存 ASP.NET Web 应用程序的配置信息（如最常用的设置 ASP.NET Web 应用程序的身份验证方式），它可以出现在应用程序的每一个目录中。当通过 ASP.NET 新建一个 Web 应用程序后，默认情况下会在根目录中自动创建一个默认的 Web.config 文件，包括默认的配置设置，所有的子目录都继承它的配置设置。如果想修改子目录的配置设置，可以在该子目录下新建一个 Web.config 文件，它可以提供除从父目录继承的配置信息以外的其他配置信息，也可以重写或修改父目录中定义的设置。

（2）创建应用程序中解决方案。下面来看一下 Visual Studio 自动产生了哪些文件。通

过查看“解决方案资源管理器”界面，可以查看相关文件，如图 1-25 所示。

项目名称的位置显示了解决方案的路径，还默认创建了 1 个文件夹和 3 个文件。

- App_Data 文件夹：用于存放数据的文件夹。
- Default.aspx：运行时产生的空白页面文档。aspx 是 ASP.NET 的文件扩展名。
- Default.aspx.cs：Default.aspx 文件的后置代码文件。
- Web.Config：在调试程序时会自动添加该文件，该文件是站点的配置文件，还可以放入一些自己用的内容。

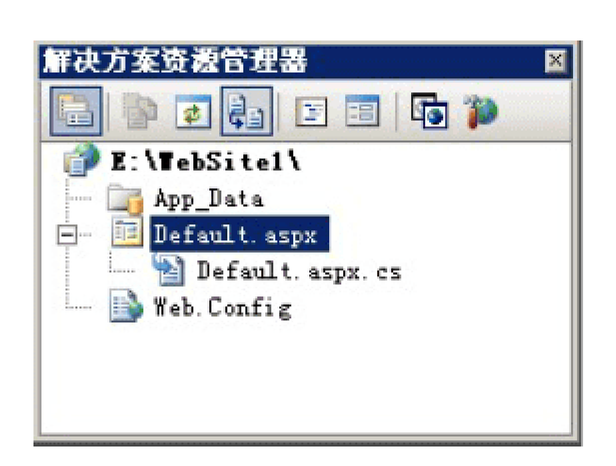

图 1-25　“解决方案资源管理器”界面

知识点小贴士

在前面学习开发桌面应用系统时，每次新建一个项目都会有一个解决方案文件，就是.sln 文件，在 ASP.NET 中，微软认为站点下的文件应该都是站点的文件，不能包含其他文件，所以在 ASP.NET 新建项目中分离了解决方案文件，将其单独地放在 C:\Documents and Settings\用户机器的用户名\My Documents\Visual Studio 2005\Projects 目录下，真正地去掉了项目文件（.csproj）。

（3）代码后置和代码内嵌。

① 代码后置。代码后置是微软的一项新技术，也是开发人员编写 ASP.NET 常用的编码方式。具体方式就是页面文件（.aspx）和代码文件（.cs）相互关联构成一个页面。一般情况下，.aspx 中没有代码、只有控件和 HTML 代码，而在.cs 文件中编写相关的代码，这样做的好处就是可以使代码和页面内容分离，使得代码清晰。

② 代码内嵌。代码内嵌时不能使用后置的.cs 文件，要完全在.aspx 文件中编写代码，只需在创建页面时取消选中的“将代码放在单独的文件中”复选框，把代码写在<% %>之间即可。

1.1.4　任务评价

标　　准	掌 握 情 况
能安装 Microsoft Visual Studio 2005	熟练□　基本□　模糊□
能安装 Microsoft SQL Server 2005	熟练□　基本□　模糊□
能安装与配置 IIS、配置基础开发环境	熟练□　基本□　模糊□
能创建 ASP.NET 应用程序	熟练□　基本□　模糊□

续表

自 我 小 结

1.1.5 任务拓展

通过学习以上内容，读者可以尝试在不同的 Microsoft Windows 操作系统上安装支持 ASP.NET 技术主要的软件，并且能够在安装好的开发环境中顺利地创建一个 ASP.NET 应用程序。

本章小结

本章主要讲解了 ASP.NET 的特色与优势，以及利用该技术开发 Web 应用程序时需要搭建的环境，介绍了如何安装 Microsoft Visual Studio 2005、Microsoft SQL Server 2005、IIS 软件，以及如何配置 Microsoft Visual Studio 2005 IDE 等内容。

自我检测

一、选择题

1．App_Data 目录用来放置（　）。

A．专业数据文件　　B．共享文件

C．被保护文件　　D．代码文件

2．以下不是 ASP.NET 特色与优势的是（　）。

A．可移植性　　B．与 ASP 应用程序的兼容性

C．扩展性与可用性　　D．可管理性

3．以下不是 Visual Studio 2005 IDE 特性的是（　）。

A．有内置 Web 服务器

B．开发程序时必须采用 IIS 作为 Web 服务器

C．网站发布变得简单

D．访问网站的方法多样

4．以下哪个版本不是 SQL Server 2005 的版本？（　）

A．SQL Server 2005 Standard Edition，标准版本

B．SQL Server 2005 Developer Edition，开发版本

C．SQL Server 2005 Professional Edition，专业版本

D．SQL Server 2005 Enterprise Edition，企业版本

5．There are three different types of file access can be chosen when create a project in

ASP.NET, they are （　）.

A．Local file system　　B．HTTP

C．FTP　　D．SMTP

二、简答题

1．简述 ASP.NET 的优势与特色。

2．简要说明需要进行 ASP.NET Web 应用程序开发时，需要安装哪些软件？如何搭建开发环境？

3．新建一个 ASP.NET 网站会自动产生哪些文档？各自的含义是什么？

第 2 章　Web 系统开发流程

技能目标

1. 能对目标系统进行需求分析。
2. 能对目标系统进行概要设计。
3. 能对目标系统进行详细设计。
4. 能掌握系统的开发流程。
5. 能编写各个开发环节中的相关文档。

相关词汇

英 文 单 词	中 文 含 义	英 文 单 词	中 文 含 义
requirement		feasibility	
specification		analysis	
documentation		module	
process		prototype	

工作任务

任务 2.1　确定系统需求
任务 2.2　进行概要设计
任务 2.3　进行详细设计

任务 2.1　确定系统需求

2.1.1　任务分析

任务目标：确定系统需求也就是通常所说的对系统进行需求分析，需求分析的效果将会在很大程度上影响后面的系统开发。需求分析阶段的任务不是具体地解决问题，而是尽可能准确地确定系统要做什么，系统应该具备怎样的功能、性能以及用户接口等要求，从而再确定系统的逻辑结构。在本任务中将主要通过分析研讨的方式来对本书中的项目案例易购商城进行需求分析，确定其系统需求。

完成标准：确定易购商城的系统需求并形成标准化的需求文档。

应用手段：采用分析研讨的方式对系统进行需求分析。

2.1.2　相关知识

2.1.2.1　软件工程

软件系统开发是一个庞大的系统工程，它涉及计算机科学、管理科学、决策科学以及系统科学等很多相关的学科，因此在其开发的过程当中，常常需要采用一些工程化的规范去协调，需要有一套规范化的方法去认识对象和开发软件，从而提高软件开发的速度、质量，同时降低软件系统的开发成本，于是就产生了软件工程的概念。

1968 年在德国召开的 NATO（North Alantic Treaty Organization，北大西洋公约组织）会议上首次提出了“软件工程”这个名词。此后，人们开始对软件开发模型、开发方法、工具与环境进行研究，提出了许多的开发模型和开发方法，以及一批 CASE（Computer Aided Software Engineering，计算机辅助的软件工程）工具和环境。

知识点小贴士

常见的软件开发模型有瀑布模型、演化模型、螺旋模型、喷泉模型以及敏捷模型等。其中瀑布模型的开发流程被认为是典型的理想化流程，对软件工程的初学者来说有很好的指导作用；而敏捷模型是近几年开始受关注的一种新的开发模型，它强调的是一种以人为核心、迭代、循序渐进的开发方式。

2.1.2.2　软件生存周期

软件产品从计划、开发、测试、运行及维护直至被淘汰的过程称为软件的生存周期。把整个软件生存周期划分为若干阶段，每个阶段都有明确的任务，就使得规模大、结构复杂和管理复杂的软件开发变得容易控制和管理。通常，软件的生存周期包括可行性分析、需求分析、设计（概要设计和详细设计）、编码、测试和运行维护 6 个活动，这些活动根据开发模型的不同将以适当的方式分配到不同的阶段去完成。如图 2-1 所示为瀑布模型中的软件生存的周期分布。

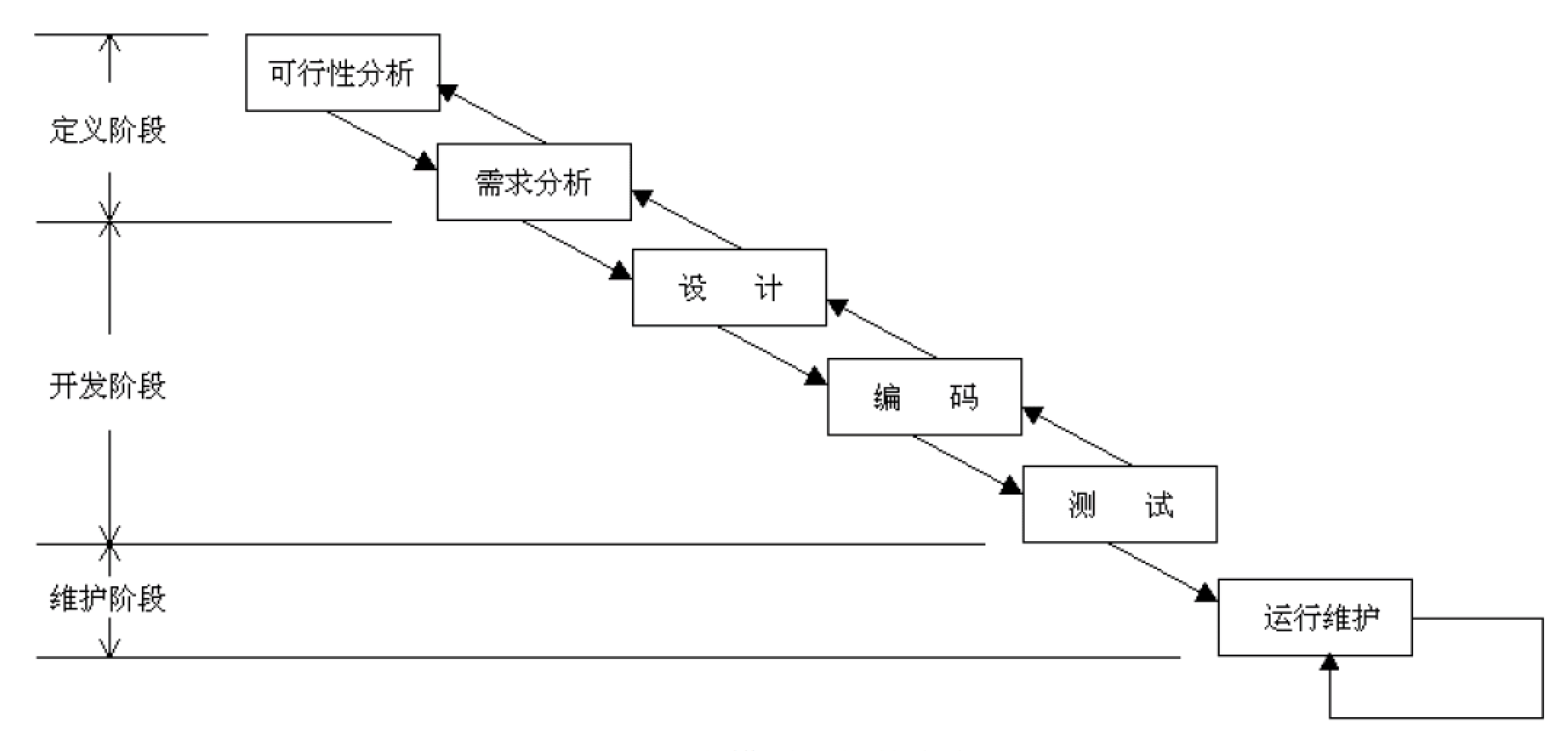

图 2-1　瀑布模型的软件生存周期

2.1.2.3 需求分析

需求分析是解决“系统要做什么”的问题，它是软件生存周期中相当重要的一个环节。由于开发人员熟悉计算机但不熟悉应用领域的业务，而用户熟悉应用领域的业务但不熟悉计算机，因此对于同一个问题，开发人员和用户之间可能存在认识上的差异。需求分析的作用就是通过开发人员与用户之间的广泛沟通，不断地让双方澄清一些模糊的概念，最终形成一个相对完整的、清晰的、一致的需求说明，进而指导以后的设计开发工作。

需求分析的过程大致上可分为 4 个步骤：问题的识别、分析与综合、编制文档和评审。

1．问题的识别

问题的识别主要是从系统的角度来理解软件，确定目标系统的综合要求，即软件的需求，同时提出这些需求实现的条件，以及需求应达到的标准，也就是从整体上明确系统要做什么，要做到什么程度。

2．分析与综合

问题分析与方案的综合是需求分析的第 2 个步骤。主要是从数据流和数据结构出发，逐步细化尽可能全面的软件功能，找出系统各元素之间的联系、接口特性和设计上的要求以及限制，分析它们是否满足功能要求、是否合理。最终，将上述的要求综合成系统的解决方案，给出目标系统的详细逻辑模型。这个步骤通常需要分析人员与用户反复磋商，直到双方都认可为止。

3．编制文档

编制文档是在第 2 步中确定下来的需求分析与综合后进行的书面化描述，从而形成通常所说的软件需求说明书。制定需求说明书时，通常会采用一定的标准来进行。这个标准可以是国家标准、行业标准或者是企业自己的标准，方便今后对文档的查阅和更新。

4．评审

为了保证需求定义的准确性和完整性，通常需要对需求分析进行评审，并提出修改意见。最终确定的需求分析才能用于设计阶段。

问题思考

在需求分析之前有一个活动叫做可行性分析，读者可以结合需求分析的相关内容思考一下在可行性分析这个环节中需要解决些什么问题？可以通过怎样的方法来解决呢？

2.1.3 任务实施

1．确定系统的综合需求

随着信息化的日益普及，网上购物的购物模式已逐渐被越来越多的人所认可，易购商城是一个基于 Web 的网上购物平台，消费者可以通过访问易购商城的网站来进行商品的浏览以及购买，同时商城的管理人员也可以通过后台的管理平台来对商城进行日常事务的管理。通过调查和分析，可以通过如下几个步骤来确定系统的需求。

（1）确定系统功能要求。通过分析当前一些主流网上商城所提供的功能，可以得到如下的一些基本功能需求。

前台：

- 商品列表的浏览（可按条件进行排序和筛选）。
- 商品详细信息的浏览。
- 商品分类导航。
- 商品购买。
- 商品搜索。
- 购物车功能。
- 用户登录。
- 用户注册。
- RSS 订阅。

后台：

- 管理登录。
- 用户管理（消费者管理和管理员管理）。
- 商品管理（商品信息的添加、修改和删除）。
- 商品分类管理（商品分类的添加、修改和删除）。
- 订单审核。

（2）确定系统界面要求。整体上要求界面有较好的亲和力，并能提供较好的可操作性，方便用户使用。经过分析，大致需要如下的界面。

前台：

- 首页（显示导航、分类、推荐商品、最新商品、热卖商品以及搜索）。
- 商品列表（能按照类别进行列表，并能按照发布日期和价格进行排序）。
- 商品详细信息显示。
- 购物车内容显示。
- 用户登录页面。
- RSS 聚合内容显示页面。

后台：

- 用户列表页面（能删除指定的用户）。
- 用户详细信息页面（能查看和修改用户的信息）。
- 用户状态修改（能修改用户的状态为正常或者无效，并可以批量修改）。
- 商品类别列表页面（能修改和删除商品分类，能重新生成用于分类导航的 XML）。
- 商品分类添加页面。
- 商品分类设定页面。
- 商品列表页面（能删除指定的商品）。
- 商品详细信息页面（能查看和修改商品信息）。
- 商品添加页面。
- 订单审核页面。
- 管理员登录页面。

（3）确定系统运行要求。服务器：CPU Intel Xeon 2.4GHz 以上，内存 2GB 以上，硬盘 SATA 500GB 以上。

Web 服务器：IIS5.1 以上。

数据库：SQL Server 2005。

.Net 框架：ASP.NET 2.0。

（4）确定开发环境。

开发工具：Visual Studio2005。

数据库：SQL Server 2005。

当然，这里列出的只是系统综合要求的部分内容，读者可以根据实际情况对其他项目进行分析和探讨，如系统的性能要求、安全性、保密性以及异常处理等。

注意事项

在实际进行需求分析时，需要开发方的分析人员和用户反复磋商才能逐一地确定各项需求，在某些情况下开发方可能需要在尽可能短的时间内根据用户的初步需求来开发一个系统原型，然后在该原型的基础上再展开与用户的协商。这里可以通过访问一些已有的网上商城，例如京东商城等，来对自己的系统做一个大致的系统需求分析，就能大致上明确易购商品的系统需求。

2．编写需求说明书

需求说明书的编写是整个需求分析的输出结果，它需要尽可能清晰和准确地描述分析阶段的结果。制定需求说明书需要按照一定的格式规范来进行，也就是通常所说的编写标准。在实际开发中各企业或者团体都会采用一套固定标准来对文档材料进行规范，国家也制定了《计算机软件产品开发文件编制指南》的标准，用于指导软件产品过程中所产生的文件的编制。表 2-1 列出了该标准中对于需求说明书的提纲要求，可以按照这个标准来制定一份易购商城系统的软件需求说明书。

表 2-1　需求说明书的提纲要求

1．引言	3.2.1　精度
1.1　编写目的	3.2.2　时间特性要求
1.2　背景	3.2.3　灵活性
1.3　定义	3.3　输入/输出要求
1.4　参考资料	3.4　数据管理要求
2．任务概述	3.5　故障处理要求
2.1　目标	3.6　其他专门要求
2.2　用户的特点	4．运行环境规定
2.3　假定的约束	4.1　设备
3．需求规定	4.2　支撑软件
3.1　对功能的规定	4.3　接口
3.2　对性能的规定	4.4　控制

知识点小贴士

需求说明书的编写也是一个反复改进的过程，在需求说明书的封面上通常会有文档版本的信息。由于需求说明书在系统开发中会指导后面的设计和实现，其准确性和完整性是非常重要的，因此，需求说明书最好也能够得到用户的认可。在某些极端情况下，需求说明书除了可以用作系统需求描述外，还可用于软件交付后，在软件功能方面与客户发生纠纷时的凭证。

2.1.4　任务评价

标　　准	掌 握 情 况
理解需求分析的作用	熟练□　基本□　模糊□
了解需求分析的一般方法	熟练□　基本□　模糊□
能对易购商城系统进行分析	熟练□　基本□　模糊□
能按标准编写需求说明书	熟练□　基本□　模糊□
自 我 小 结	

2.1.5　任务拓展

读者可以通过对一些主流的网上商城进行功能分析，来完善本节中的需求分析，然后根据国家标准中对软件需求说明书的要求，通过分析、讨论以及参阅等形式来完成需求说明书的编写。

任务 2.2　进行概要设计

2.2.1　任务分析

任务目标：在明确了易购商城的系统需求后，下一步就要进入到系统的设计阶段，解决目标系统“怎么做”的问题。系统设计的主要目的就是为系统制定蓝图，在各种技术和实施方法中权衡利弊，精心设计，合理使用各种资源，最终勾画出新系统的详细设计方案。在该任务中，需要根据前面需求分析阶段所产生的需求说明书来进行易购商城系统的第 1 部分概要设计。

完成标准：能够根据系统需求说明书来进行概要设计，并完成数据库的设计以及生成相应的设计文件。

应用手段：采用分析研讨等方式进行概要设计。

2.2.2 相关知识

2.2.2.1 系统设计

系统设计的主要内容包括新系统总体结构设计、代码设计、输出设计、输入设计、处理过程设计、数据存储设计、用户界面设计和安全控制设计等。一般将系统设计大体上分为概要设计和详细设计两个步骤。

注意事项

系统设计是一个相当复杂和细致的过程，它要求设计人员除了具备良好的软件开发能力外，还要具备一定的设计经验。本书中侧重阐述软件系统开发的流程，而对于每个阶段涉及的一些详细的设计方法没有进行展开，有兴趣的读者可以参考相关的软件工程书籍。

2.2.2.2 概要设计

概要设计的目的是解决实现需求的程序模块设计问题。在这个阶段中，开发人员需要确定软件系统的结构，进行模块划分，确定每个模块的功能、接口以及模块之间的调用关系。同时，这个阶段也需要对全局数据结构进行设计，详细设计阶段还应该对局部数据结构进行设计。在概要设计阶段一般需要完成如下的几项工作。

1．制定规范

制定规范用于协调在设计阶段各部分的工作，使参与人员能遵守一个标准，如设计方法、设计的时间计划以及文档编写标准等。

2．系统结构的总体设计

系统结构的总体设计的主要任务是将整个系统按功能划分成模块，确定每个模块的功能，以及每个模块之间的调用关系和接口等。

知识点小贴士

模块的划分是一项非常重要的工作，因为模块不能简单地按照在需求分析阶段生成的功能来划分。系统在对模块进行划分时，应使模块具有独立性，即应使模块功能单一，与其他模块没有或尽可能少的相互作用与相互依赖，还应使模块的接口简单。

3．数据结构设计

数据结构设计是指在需求分析阶段对系统数据要求所做的分析的基础上，进一步设计数据库。包括模式设计、子模式设计、完整性和安全性设计以及优化等。

4．编写概要设计文档

编写概要设计文档主要包括概要设计说明书、数据库设计说明书、用户手册以及修订测试计划等。

5．评审

评审是对概要设计工作进行评议和审查，保证阶段性工作的正确性。

2.2.3　任务实施

1．功能分解

通过对需求功能进行分析，进一步对功能进行细化，将功能划分为若干个模块。在概要设计中，可以采用模块结构图来对模块及其关系进行描述。

知识点小贴士

模块结构图是结构化设计中描述系统结构的图形工具。它严格地定义了模块的名字、功能和接口，同时还在模块结构图上反映出结构化设计的思想。模块结构图由模块、调用、数据、控制和转接等符号组成，如图 2-2 所示。

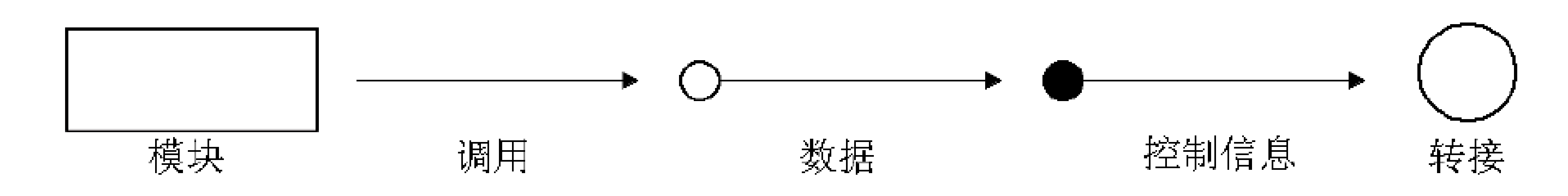

图 2-2　模块结构图基本符号

这里以商品分类管理这个功能模块为例，来介绍如何将其细分为若干个功能模块，然后用模块结构图的方式进行表达。按照用户的需要，商品分类管理应该包括的子功能有：商品分类查看、商品分类删除、商品分类修改以及商品分类添加，因此，可以将这些子功能作为商品分类管理中的功能模块，于是可以得到如图 2-3 所示的模块结构图。

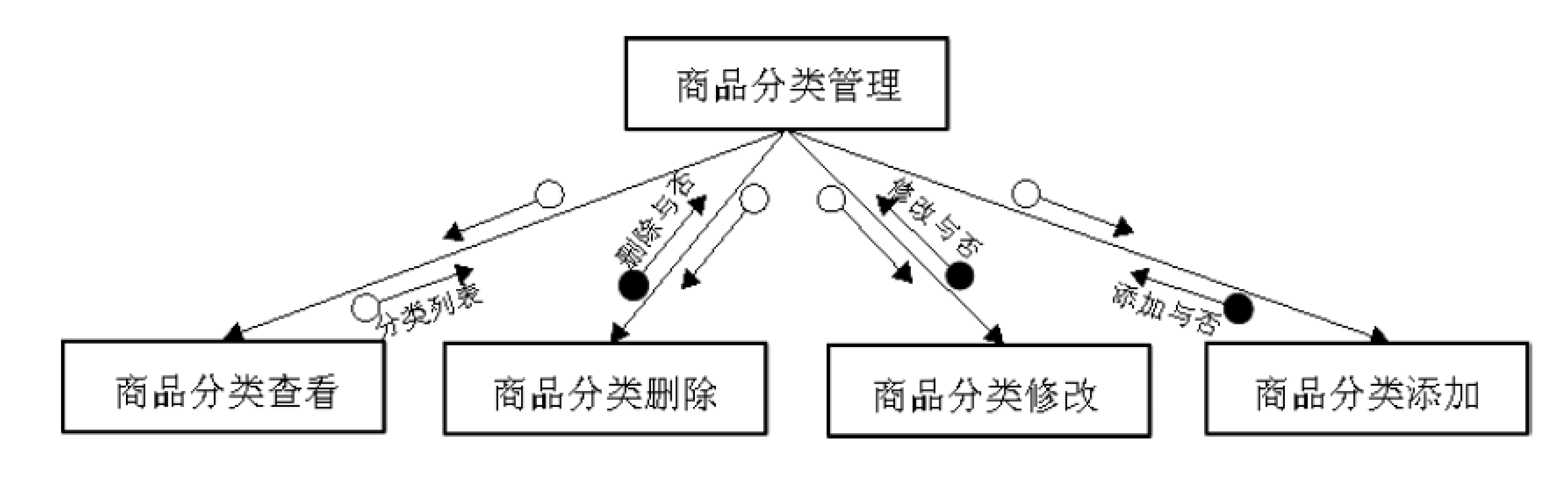

图 2-3　模块结构图

通过进一步分析还可以得出，商品分类查看除了查看商品分类的列表外还可以查看某一个商品分类的信息；同时，商品分类的修改和商品分类的添加功能都需要对数据库中已有的商品分类信息进行比对，看是否已经有同样的商品分类存在。因此，图 2-3 所示的模块结构图还能够进一步细化为如图 2-4 所示的样子。

这样就可以直观地看出某一个功能包含哪些模块、这些模块之间的关系，以及模块与模块之间需要传递怎样的数据或者信息。

2．数据结构设计

在对系统的功能进行分析之后，就可以开始着手数据库的设计，关于数据库设计的一些方法和技术大家可以参阅一些数据库方面的书籍。通过对系统功能的分析以及模块化，可以得出本系统数据库应该具备的实体有：商品表（Items）、商品分类表（Categories）、商品品牌表（Brands）、订单表（Orders）、订单明细表（OrderItems）、用户表（Users）、

用户状态表（UserRoles）以及用户权限表（UserStates）。

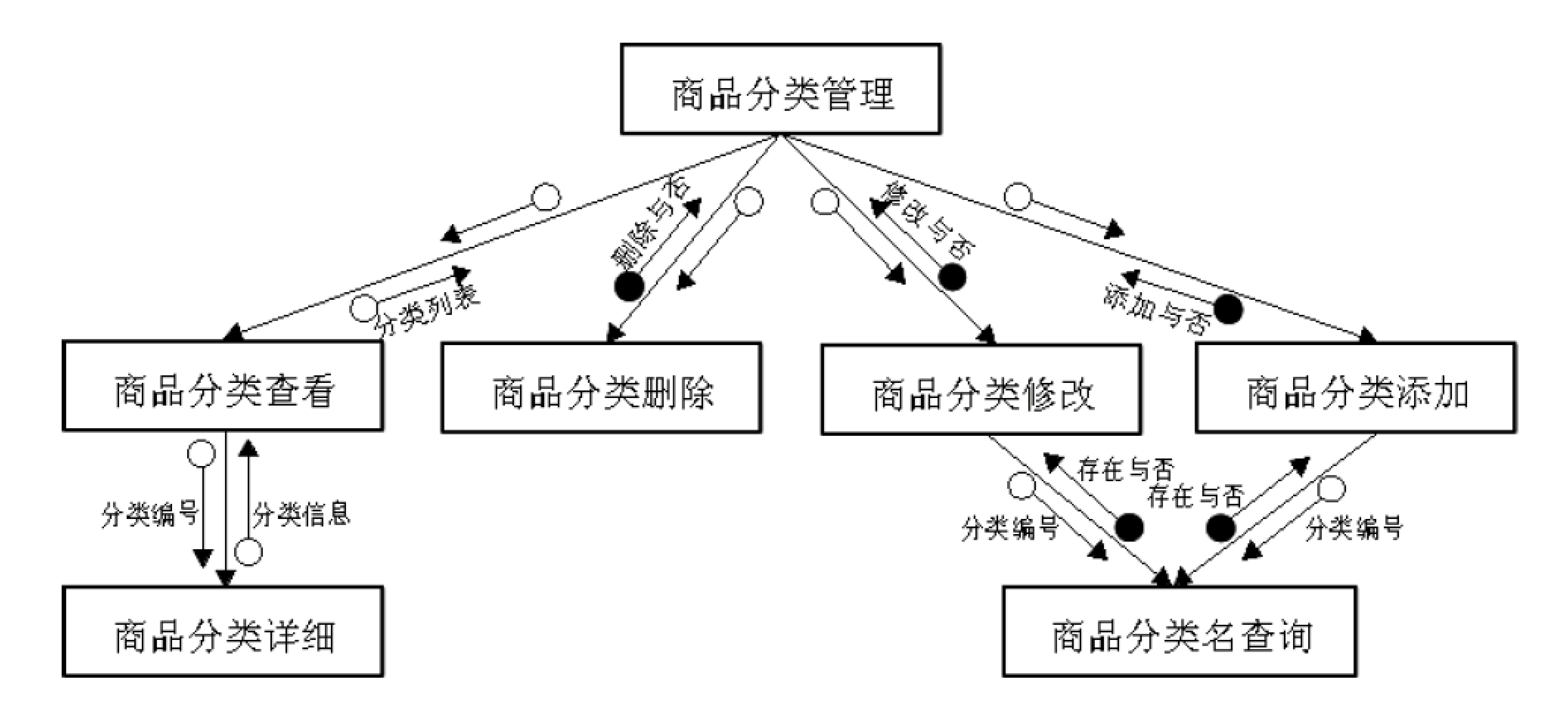

图 2-4　细化的模块结构图

在设计时，首先要设计上述的数据库实体，然后还要根据系统的功能来设计实体的字段，以及实体与实体之间的关系，一般可以采用 E-R 图来对它们进行描述，例如，商品表它应该具备的字段有商品编号（Id）、商品名称（Title）、价格（Price）、商品描述（Description）、品牌编号（BrandId）、商品分类编号（CategoryId）以及发布日期（PublishDate）等信息，如图 2-5 所示就是采用 E-R 图来对商品实体进行描述。

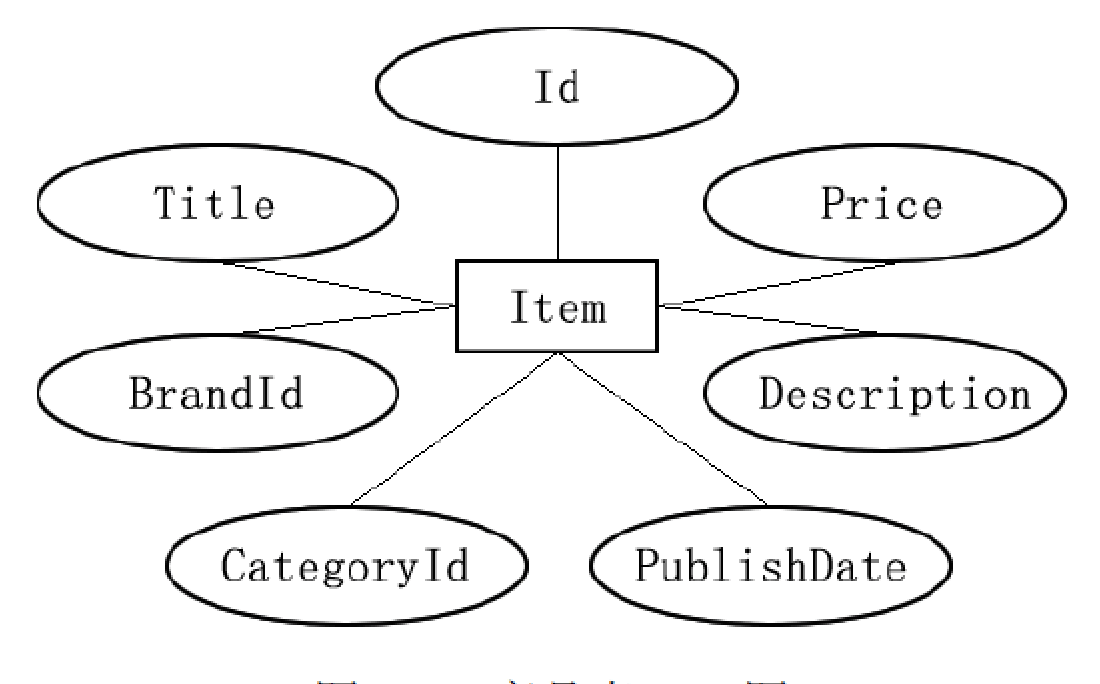

图 2-5　商品表 E-R 图

商品表中两个字段 CategoryId 和 BrandId 分别是与分类表和品牌表这两个实体发生关系的，设计数据库时也需要将这些实体与实体之间的关系体现出来，如图 2-6 所示。最后需要把系统中所有的实体、字段以及它们之间的关系都在 E-R 图中体现出来，并且需要设计字段的类型。

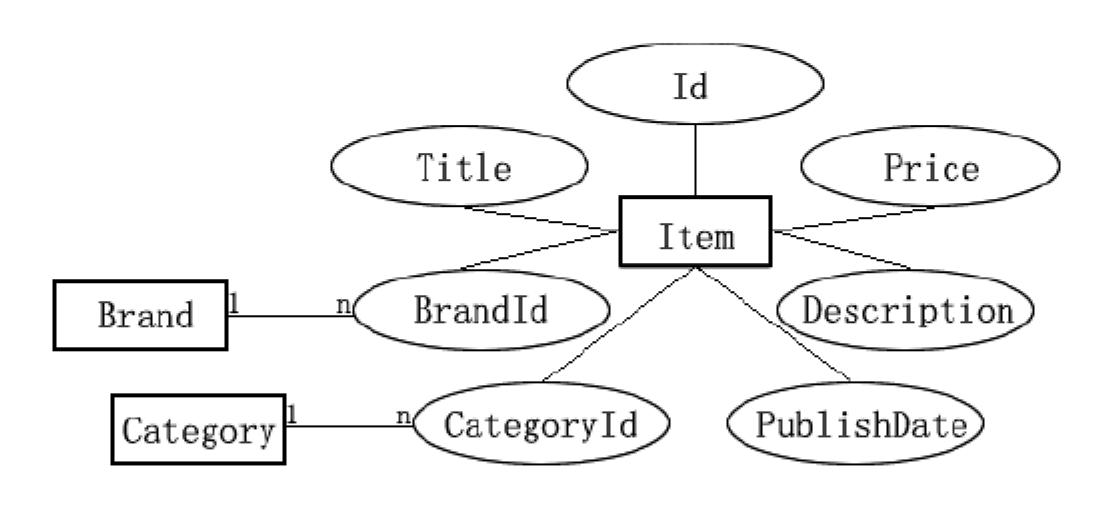

图 2-6　体现实体之间关系的 E-R 图

3. 编写概要设计说明书

完成了模块设计以及数据结构设计后，就可以开始编写概要设计说明书了。概要设计说明书的内容可以参考表 2-2《计算机软件产品开发文件编制指南》中概要设计说明书的提纲。

表 2-2　概要设计说明书提纲

1. 引言	3.2　外部接口
1.1　编写目的	3.3　内部接口
1.2　背景	4. 运行设计
1.3　定义	4.1　运行模块组合
1.4　参考资料	4.2　运行控制
2. 总体设计	4.3　运行时间
2.1　运行环境	5. 系统数据结构设计
2.2　基本设计概念和处理流程	5.1　逻辑结构设计要点
2.3　结构	5.2　物理结构设计要点
2.4　功能需求与程序的关系	5.3　数据结构设计要点
2.5　人工处理过程	6. 系统出错处理设计
2.6　尚未解决的问题	6.1　出错信息
3. 接口设计	6.2　补救措施
3.1　用户接口	6.3　系统维护设计

问题思考

其实在概要设计阶段有很多的设计内容，而且还有很多的内容需要通过图来进行表达，那么在实际开发中究竟需要对哪些内容进行设计？要采用哪些图以及按照怎样的一个步骤来进行呢？

2.2.4　任务评价

标　　准	掌 握 情 况
理解系统设计的作用	熟练□　基本□　模糊□
理解概要设计的作用	熟练□　基本□　模糊□
能进行系统结构总体设计	熟练□　基本□　模糊□
能进行数据结构设计	熟练□　基本□　模糊□
能编写概要设计说明书	熟练□　基本□　模糊□
自 我 小 结	

2.2.5 任务拓展

按照本节所阐述的方法，读者可对前面在需求分析中提出的功能进行分解，并根据功能完成数据库的设计。在此基础上，根据国家标准中概要设计说明书的要求，通过分析和讨论等方式，在系统结构总体设计和数据结构设计的基础上完成概要设计说明书的编写。

任务 2.3 进行详细设计

2.3.1 任务分析

任务目标：概要设计的结果还不能直接地用于开发工作，还需要对每个模块进行进一步的细化，这个细化的过程就是详细设计。在概要设计中，设计的对象是整个系统，而在详细设计中，设计的对象可以是划分好的模块。详细设计的工作是在概要设计的基础上来开展的，其设计的结果可以直接用于指导开发，一个好的详细设计可以分配给不同的开发小组分别进行开发，从而提高开发效率。在这个任务中，同样将参照详细设计的一般方法来对易购商城系统进行设计。

完成标准：能够对概要设计中产生的各模块进行详细设计并生成相关的文件。

应用手段：采用分析研讨的方式进行详细设计。

2.3.2 相关知识

2.3.2.1 详细设计

详细设计是为模块实现细化设计。在概要设计时已确定了每个模块的功能和接口，在详细设计的过程中需要决定每个模块的实现算法，并精确地表达这些算法。详细设计的主要任务包括如下几项。

（1）为每个模块进行详细的算法设计。可采用图形、表格或语言等工具对每个模块的处理过程进行描述。

（2）对模块内的数据结构进行设计。概要设计是对整个系统进行数据结构的设计，而详细设计是以模块为单位进行更详细的数据结构设计。

（3）数据库物理设计。数据库物理设计即确定数据库的物理结构，实现数据库。

（4）其他设计。根据系统的类型，还可以有选择地进行如人机交互或者输入/输出等内容的设计。

（5）编写详细设计说明书。按照一定的规范要求，将分析的结果编写成详细设计说明书。

（6）评审。对处理过程的算法和数据库的物理结构都要进行评审。

2.3.2.2　结构化程序设计

详细设计的主要方法是结构化程序设计。结构化程序设计采用的是自顶向下逐步求精的设计方法和单入口单出口的控制结构。结构化程序设计方法是将任何程序设计为几种基本的控制结构：顺序结构、分支结构和循环结构等，这些结构如图 2-7 所示。

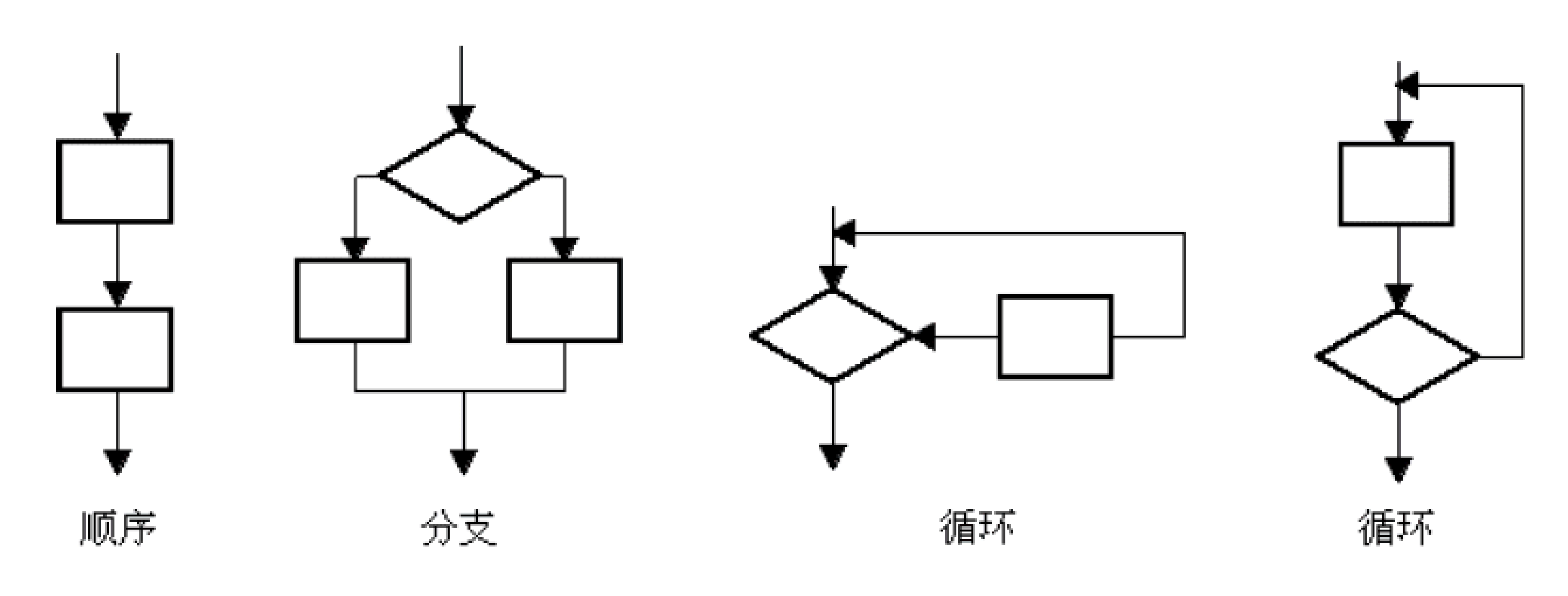

图 2-7　结构化程序设计基本控制结构

使用结构化程序设计的优点是可以用统一的表述方式来对模块的功能进行更详细的描述，在后面的任务中也将采用这种方法来进行模块的详细设计。

知识点小贴士

在详细设计中还有一些不同类型的描述工具，例如，图形描述工具、语言描述工具和表格描述工具等。

常用的图形描述工具有：程序流程图、盒图和问题分析图等。

常用的语言描述工具有：算法描述语言（PDL）和结构化语言（英语）等。

常用的表格描述工具有：判断表等。

2.3.3　任务实施

1．处理过程设计

在处理过程设计中，将对用户登录模块进行设计。登录模块是最常用的模块之一，它的主要作用是对用户输入的用户名和密码进行核对。若用户名和密码能够与保存在数据库中的数据匹配则允许登录，否则就不允许登录。经过分析可以得出如图 2-8 所示的处理流程，也可以采用同样的方法来处理其他模块的流程设计。

2．其他设计

对于图 2-8 所示的流程登录模块，开发者还可以对其进行输入/输出以及界面的大致设计。登录模块的输入数据应该是两个字符串用户名和密码，并且这两者都不能为空。而根据处理流程的设计，它的输出应该有两种情况：一种情况是登录成功，当用户输入的用户名和密码与数据库中的信息相匹配时可认定为登录成功，这时候需要进行的处理是引导用户进入登录成功页面并做相应的后台处理；另一种情况是登录失败，即用户输入的数据和数据库中保存的数据不能匹配，这时候用户可以选择重新输入用户名和密码或者离开。

而对于界面设计，根据输入数据的要求，登录模块应该具备两个用来输入用户名和密

码的接口，当然至少还需要一个用于提交的按钮，开发小组在讨论时可以在白纸上或者白板上画出如图 2-9 所示的登录界面的草图。

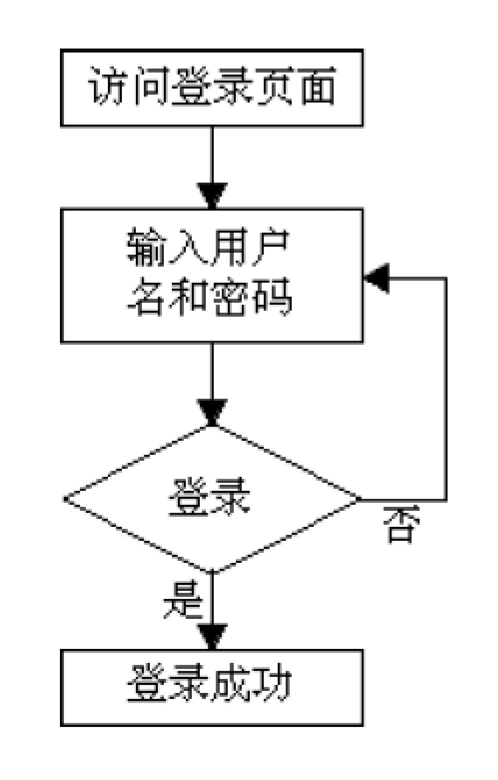

图 2-8　登录模块的处理流程

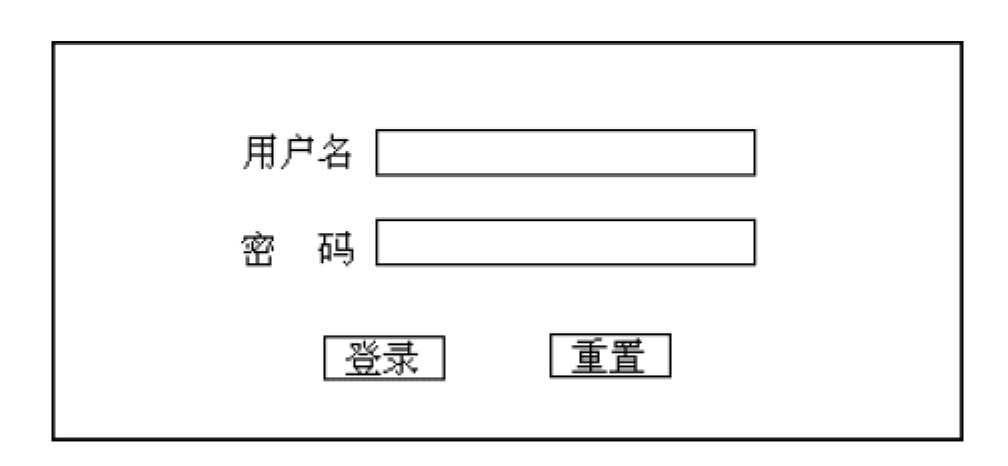

图 2-9　登录界面草图

知识点小贴士

在实际的设计过程中，设计人员和小组经常会用笔直接在白纸或者白板上画出流程图等图形描述，或者直接画出设计界面供开发小组成员参考，并将经过大家讨论确认的描述写入文档。当然，有些时候如果手绘的图形足够清楚的话，直接对其进行拍照后放入文档也不失为一种高效的办法。

3．编写详细设计说明书

详细设计说明书是对每个模块进行的详细设计说明，一份好的详细设计说明书可以直接用于指导系统开发。详细设计说明书经过审核确认后，即可开始后面的系统开发部分，本书从第 3 章起就开始进行系统的开发实现。表 2-3 是《计算机软件产品开发文件编制指南》中的详细设计说明书的提纲。

表 2-3　详细设计说明书提纲

1．引言	3.5　输出项
1.1　编写目的	3.6　算法
1.2　背景	3.7　流程逻辑
1.3　定义	3.8　接口
1.4　参考资料	3.9　存储分配
2．程序系统的组织结构	3.10　注释设计
3．程序 1（结构符）设计说明	3.11　限制条件
3.1　程序描述	3.12　测试设计
3.2　功能	3.13　尚未解决的问题
3.3　性能	4．程序 2（标识符）设计说明书
3.4　输入项	

2.3.4　任务评价

标　　准	掌 握 情 况
理解详细设计的作用	熟练□　基本□　模糊□
能对模块进行流程设计	熟练□　基本□　模糊□
能对模块进行输入输出设计	熟练□　基本□　模糊□
能对模块进行界面设计	熟练□　基本□　模糊□
能编写详细设计说明书	熟练□　基本□　模糊□
自 我 小 结	

2.3.5　任务拓展

参照前面的步骤，完成系统各模块的详细设计，并能用图的方式对主要的功能模块进行描述。通过分析和讨论，在详细设计的基础上编写各模块的详细设计说明书。

本章小结

软件的开发是一个系统工程，它要求开发人员能够按照一定的开发流程来进行协调和工作，其中的每个步骤都会对整个软件系统的开发产生影响。本章以易购商城为例，对软件系统的开发流程作了一个简单的阐述，其目的主要是希望读者能够对软件系统的开发流程有个整体上的认识、能够理解软件工程的基本概念，同时了解软件生命周期的几个主要阶段（需求分析、概要设计、详细设计、编码实现、测试及运行维护）以及在各个阶段中需要完成的任务。

自我检测

一、选择题

1．需求分析中，开发人员要通过用户解决的最重要的问题是（　）。

A．要让软件做什么　　　　B．要给软件提供哪些信息

C．要求软件工程效率怎样　　　　D．要让软件具有何种结构

2．软件重用的单位是（　）。

A．系统　　　　B．性能

C．软件模块　　　　D．功能

3．软件测试的目的是（　）。

A．评估软件的质量　　B．发现软件的错误

C．证明软件是对的　　D．发现软件的所有错误

4．“软件危机”是指（　）。

A．计算机病毒的出现

B．利用软件系统进行经济犯罪的活动

C．人们过分依赖计算机软件系统

D．软件开发和软件维护中出现的一系列问题

5．软件的生命周期是指（　）。

A．软件系统开始研制到软件系统投入运行

B．软件系统投入运行到软件系统被废弃

C．软件系统开始研制到软件系统开发完成

D．软件系统开始研制到软件系统被废弃

6．During the detail designing we usually use a method for design, which is （　）.

A．structured programming　　B．model designing

C．structured designing　　D．flow chart

二、简答题

1．简述软件系统开发的一般流程。

2．一般采用什么步骤来进行需求分析？

3．简述概要设计和详细设计的过程。

第 3 章　搭建系统框架

技能目标

1．能理解三层结构在 ASP.NET 中的应用。
2．能实现三层结构下的用户登录和注册。

相关词汇

英 文 单 词	中 文 含 义	英 文 单 词	中 文 含 义
modify		retrieve	
execute		scalar	
command		register	
framework		layer	

工作任务

任务 3.1　设计系统框架
任务 3.2　设计模型层
任务 3.3　设计数据访问层
任务 3.4　设计业务逻辑层
任务 3.5　设计表示层

任务 3.1　设计系统框架

3.1.1　任务分析

任务目标：搭建系统框架，是整个开发工作中首先需要解决的问题。本任务中，需要搭建 eBuy 网上商城系统的整体结构，包括模型层、数据访问层、业务逻辑层和表示层的基本框架，并且需要完成用户管理模块的设计。其他部分的设计参考用户管理模块，读者自行设计。

完成标准：按三层架构来搭建系统的总体框架。

应用手段：分层创建系统中的各个项目。

3.1.2 相关知识

通常意义上的三层架构是指将整个业务应用划分为表示层（UI）、业务逻辑层（BLL）和数据访问层（DAL），如图 3-1 所示。

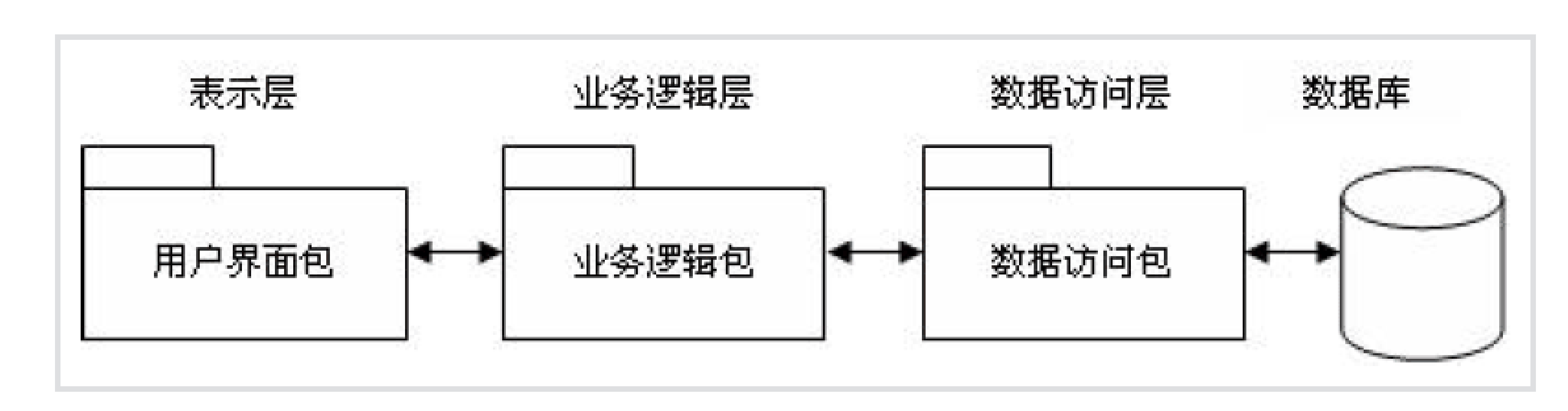

图 3-1 三层结构

所谓“三层体系结构”，是指在客户端与数据库之间加入了一个“中间层”，也叫“组件层”。但这里所说的三层体系，不是指物理结构上的层次，不是简单地放置三台机器就能形成三层体系结构，也不是仅仅有 B/S 应用才是三层体系结构，而是指逻辑上的三层，哪怕这三个层是放置在同一台机器上。

三层体系的应用程序将业务规则、数据访问和合法性校验等工作放在中间层进行处理。通常情况下，客户端不直接与数据库进行交互，而是通过 COM/DCOM 通信与中间层建立连接，再经由中间层与数据库进行交互。

3.1.2.1 数据访问层

数据访问层有时也称为“持久层”，其功能主要是负责数据库的访问，例如，访问数据库系统、二进制文件、文本文档或 XML 文档。简单地说就是实现对数据表的 Select、Insert、Update 和 Delete 的操作，不做业务逻辑的判断。

数据访问层项目一般命名为 DAL 或者解决方案名+DAL。在本书的项目中，将其命名为 eBuyShopDAL。

3.1.2.2 业务逻辑层

业务逻辑层（Business Logic Layer）用于做一些有效性验证的工作，以更好地保证程序运行的健壮性。例如完成数据添加、修改和查询业务等；不允许指定的文本框中输入空字符串，数据格式是否正确以及数据类型验证；用户权限的合法性判断等。通过这些判断来决定是否将操作继续向后传递，从而保证程序的正常运行。

业务逻辑层在体系架构中的位置非常关键，它处于数据访问层与表示层之间，起到了数据交换中承上启下的作用。因为层是一种弱耦合结构，层与层之间的依赖是向下的，底层相对于上层而言是“无知”的，改变上层的设计对其调用的底层没有任何影响。如果在分层设计时，遵循了面向接口设计的思想，那么这种向下的依赖也应该是一种弱依赖关系。因而在不改变接口定义的前提下，理想的分层式架构，应该是一个支持可抽取、可替换的“抽屉”式架构，正因为如此，业务逻辑层的设计对于一个支持可扩展的架构尤为关键，因为它扮演了两个不同的角色。对于数据访问层而言，它是调用者；对于表示层而言，它

却是被调用者。依赖与被依赖的关系取决于业务逻辑层上的角色，如何实现依赖关系的解耦，则是除了实现业务逻辑功能之外留给设计者的任务。

业务逻辑层项目一般命名为“BLL”或者“解决方案名+BLL”，在本书的项目中命名为 eBuyShopBLL。

3.1.2.3　表示层

表示层位于最外层（最上层），离用户最近，用于显示数据和接收用户输入的数据，为用户提供一种交互式操作的界面。许多项目中，表示层往往是项目成败的关键，而这也是容易被程序员所忽视的。

在 ASP.NET 中，表示层就是整个 Web 站点，具体的内容要根据需求来定。本书所做的 eBuyShop 项目包括很多功能，但在本章仅设计用户管理模块，包括用户的登录、注册和管理等方面。

三层体系结构的优点如下：

（1）开发人员可以只关注整个结构中的某一层。

（2）很容易用新的实现来替换原有层次的实现。

（3）可以降低层与层之间的依赖。

（4）有利于标准化。

（5）有利于各层逻辑的复用。

三层体系结构的缺点如下：

（1）降低了系统的性能。如果不采用分层式结构，很多业务可以直接访问数据库，以此获取相应的数据，但现在却必须通过中间层来完成。

（2）有时会导致级联的修改。这种修改主要体现在自上而下的方向。例如，需要在表示层中增加一个功能，为保证其设计符合分层式结构，需要在相应的业务逻辑层和数据访问层中都增加相应的代码。

一个项目是不是应该采用三层/多层设计，其判断标准是它是不是真的有需要。实际上，大部分程序开个 WebApplication 就足够了，完全没必要做得这么复杂。因此，多层结构主要用于解决真正复杂的项目需求。

由于三层之间的数据都需要通过传递实体对象来达到目的，一般都会单独创建一个 Model 项目，它包含与数据库表相对应的实体类，三层都要使用该项目的类，因此，将其独立出来，称为模型层。

一般模型层的项目名称为 Model 或者 Models，也可以用“解决方案名+Models”表示，在本书的项目中命名为 eBuyShopModels。模型层中的实体类一般与所对应的表名一致。

3.1.3　任务实施

1．创建表示层项目

打开 Visual Studio 2005，选择“文件”|“新建”|“网站”命令，打开如图 3-2 所示的“新建网站”对话框，完成网站的创建。

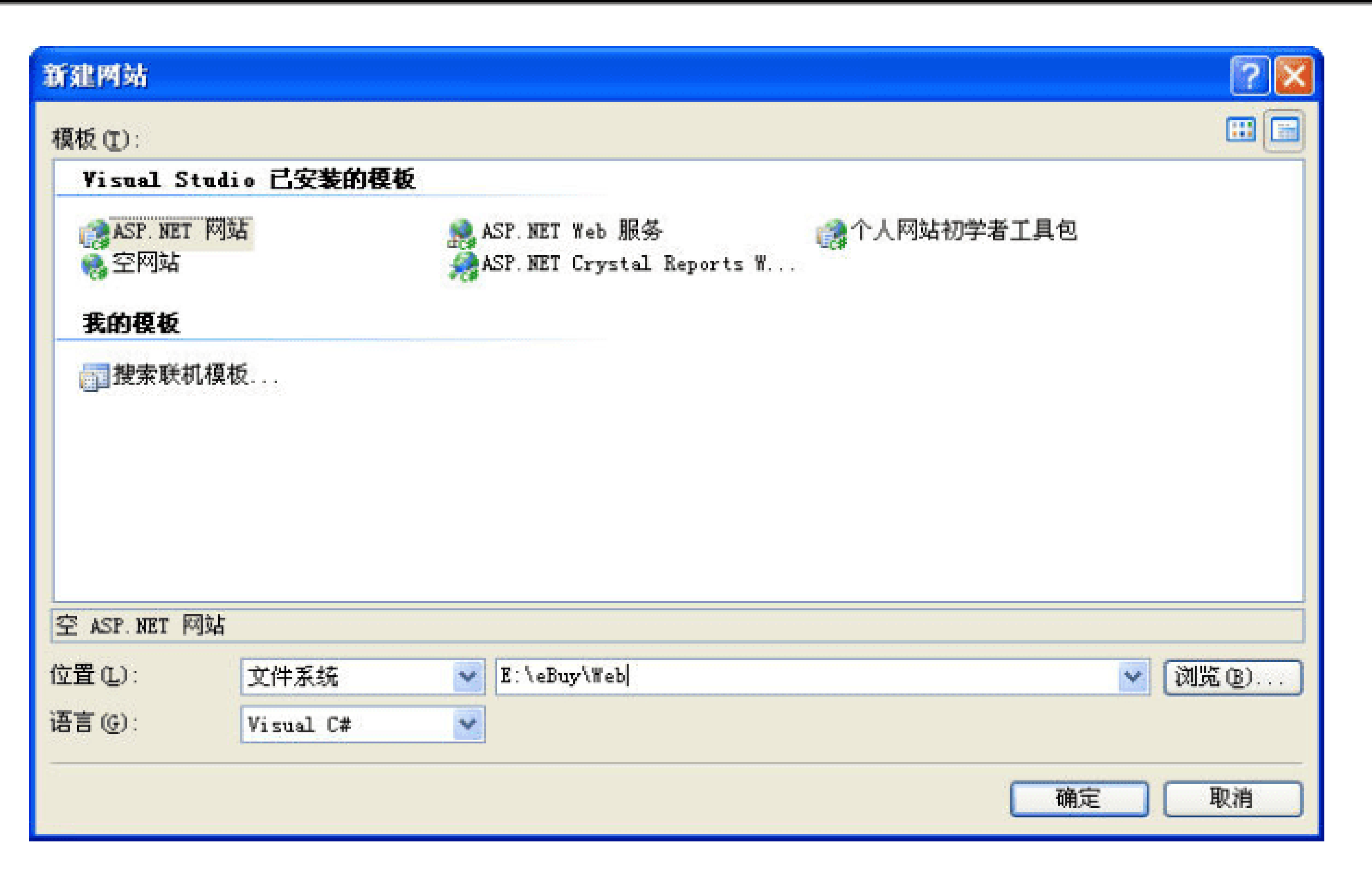

图 3-2　“新建网站”对话框

为了方便对整个项目进行管理，需要将项目的所有文件都放在一个名为“eBuy”的文件夹中，现在创建的是 Web 站点部分，也就是表示层部分。

通过前面的操作，已经在 Visual Studio 的 IDE 中创建了一个名为“Web”的网站。通过“解决方案资源管理器”查看随着网站一起产生的一些文件或文件夹，如图 3-3 所示。

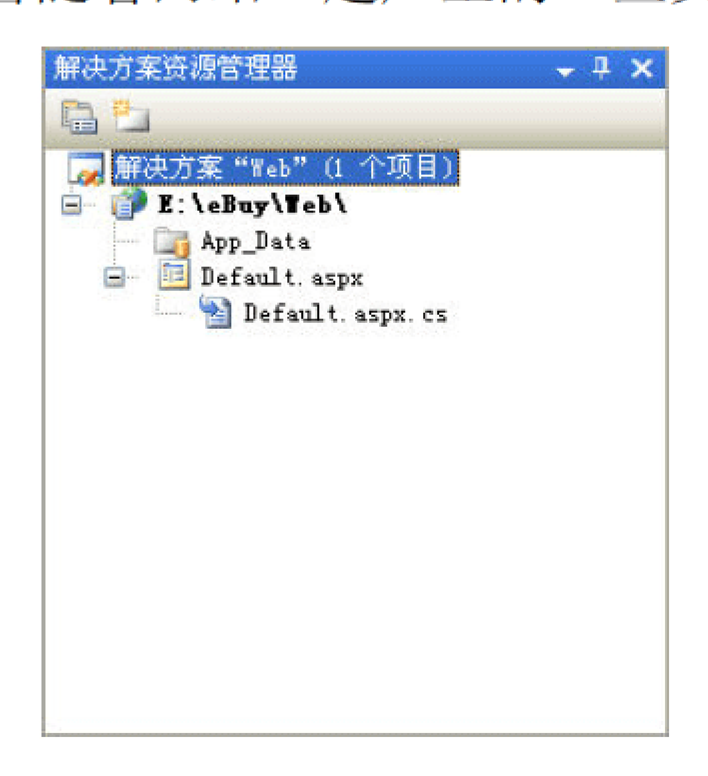

图 3-3　解决方案资源管理器

从图 3-3 中可以看到，网站项目的名称显示了项目当前所在的位置，随着项目的生成同时也产生了一个名为“App_Data”的文件夹和一个名为“Default.aspx”的页面文件以及它的代码后置文件“Default.aspx.cs”。

知识点小贴士

App_Data 文件夹一般用来存放数据文件，例如保存数据的 XML 文件就可以放在其中。Default.aspx 是自动产生的一个空白页面，采用 ASP.NET 开发而成，其后缀名是.aspx。通常情况下，可以根据网页的后缀名来判断其所在网站采用的开发技术，常见的开发技术有.PHP、.ASP、.ASPX 和.JSP。

Default.aspx.cs 是 Default.aspx 的代码后置文件。在 ASP.NET 中，客户端代码和服务器段代码可以被很好地分离，而后台的服务器段代码就保存在这个文件当中。

同样，将项目解决方案名改为“eBuyShop”，并重新生成解决方案，通常情况下，解

决方案文件保存在 C:\Documents and Settings\Administrator\My Documents\Visual Studio 2005\Projects\Web 文件夹下，将其复制到 eBuy 文件夹中。整理后的项目文件夹窗口如图 3-4 所示。

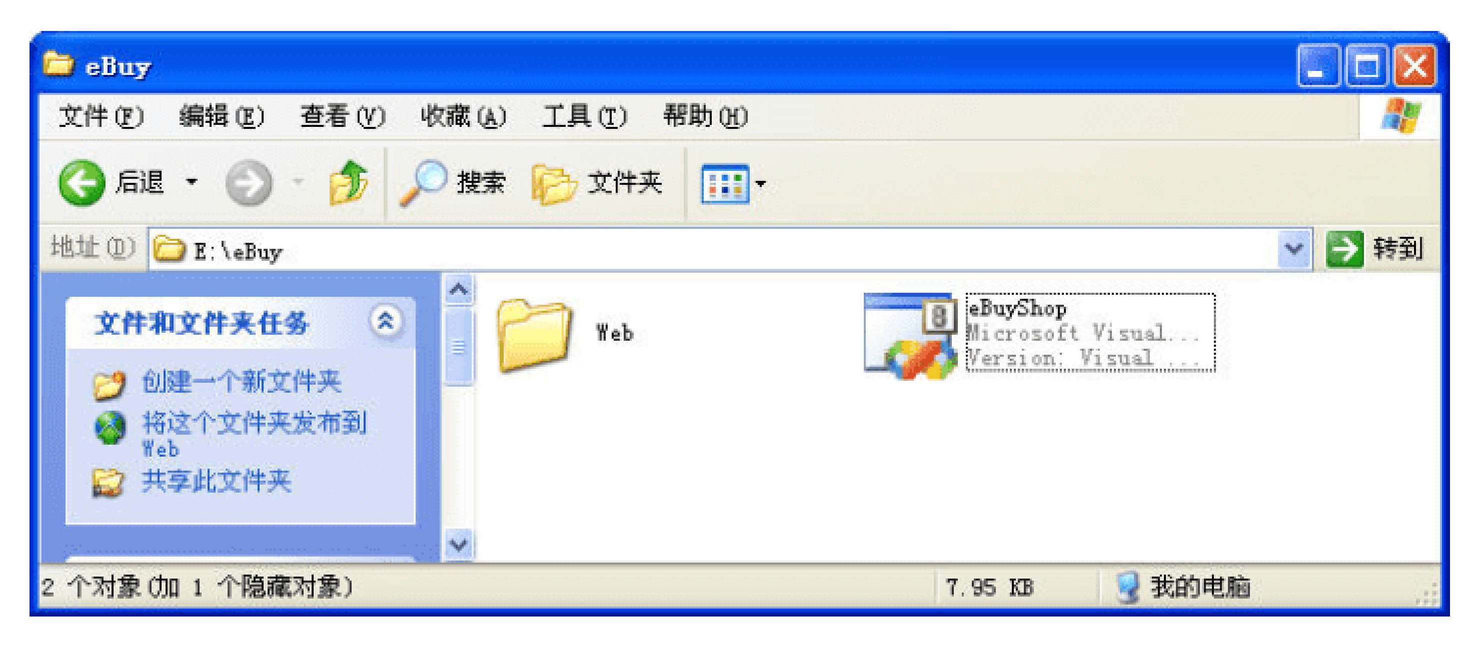

图 3-4　整理后的项目文件夹窗口

2．创建业务逻辑层项目

右击“解决方案资源管理器”中的“解决方案‘eBuyShop’”，在弹出的快捷菜单中选择“添加”|“新建项目”命令，即可打开如图 3-5 所示的“添加新项目”对话框。在该对话框中将项目名改为“eBuyShopBLL”，设定好存放位置后，单击“确定”按钮，即可完成业务逻辑层项目的创建。

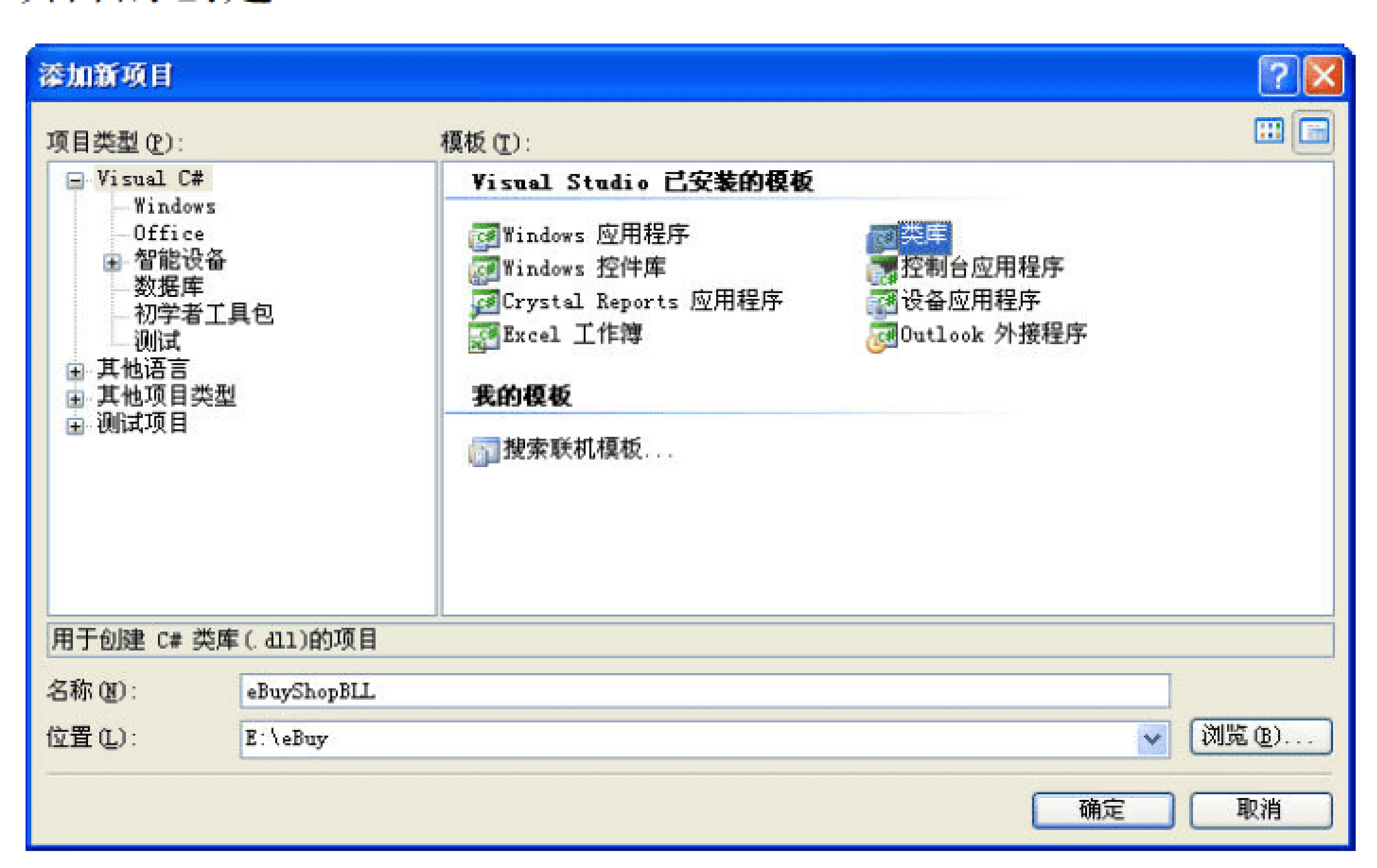

图 3-5　创建业务逻辑层项目

3．创建数据访问层项目

采用和创建业务逻辑层项目类似的方法，创建一个数据访问层项目，并将其名为“eBuyShopDAL”，如图 3-6 所示。

4．创建模型层项目

采用和创建业务逻辑层项目类似的方法，创建一个模型层项目，并将其名为“eBuyShopModels”，如图 3-7 所示。

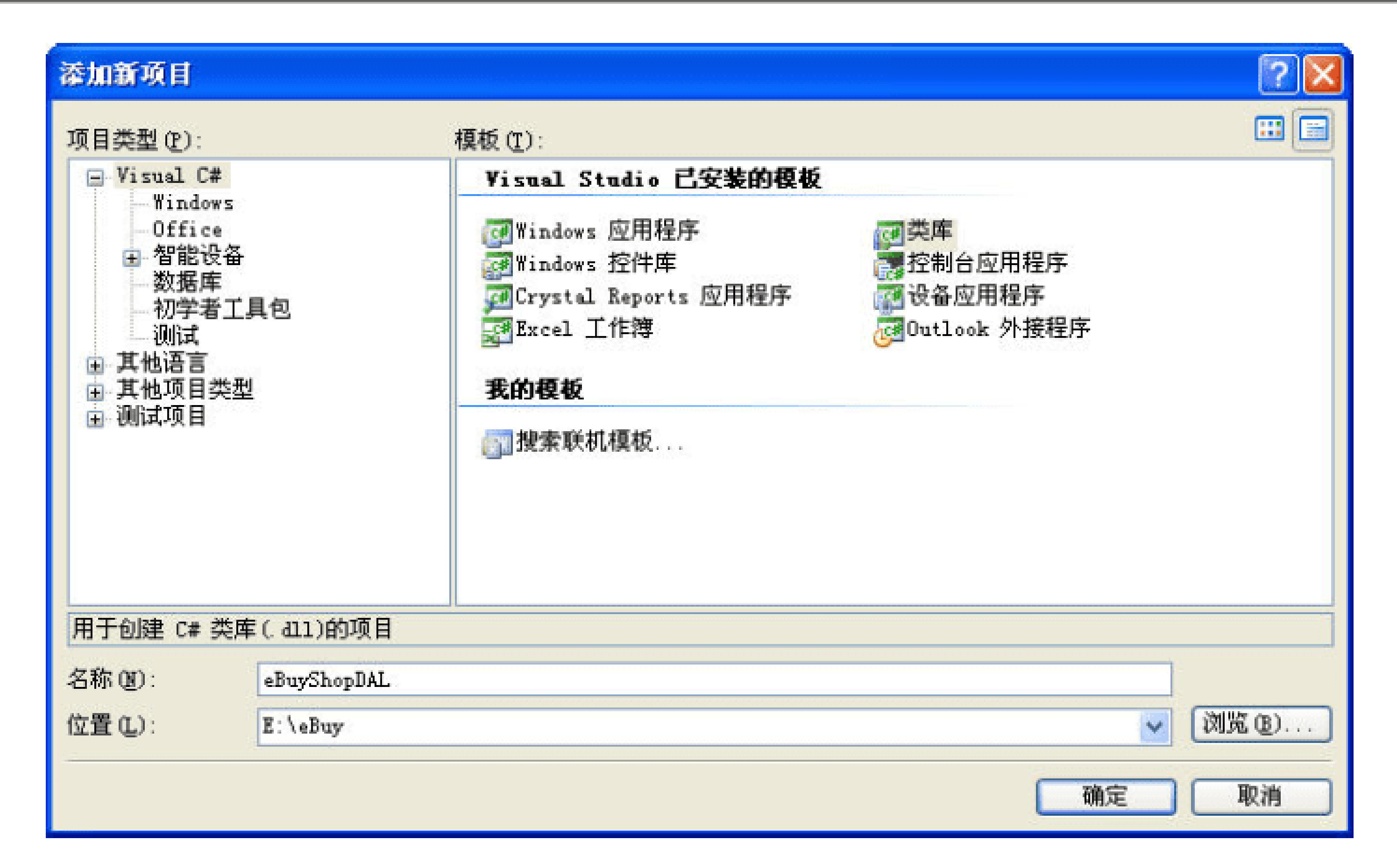

图 3-6 创建数据访问层项目

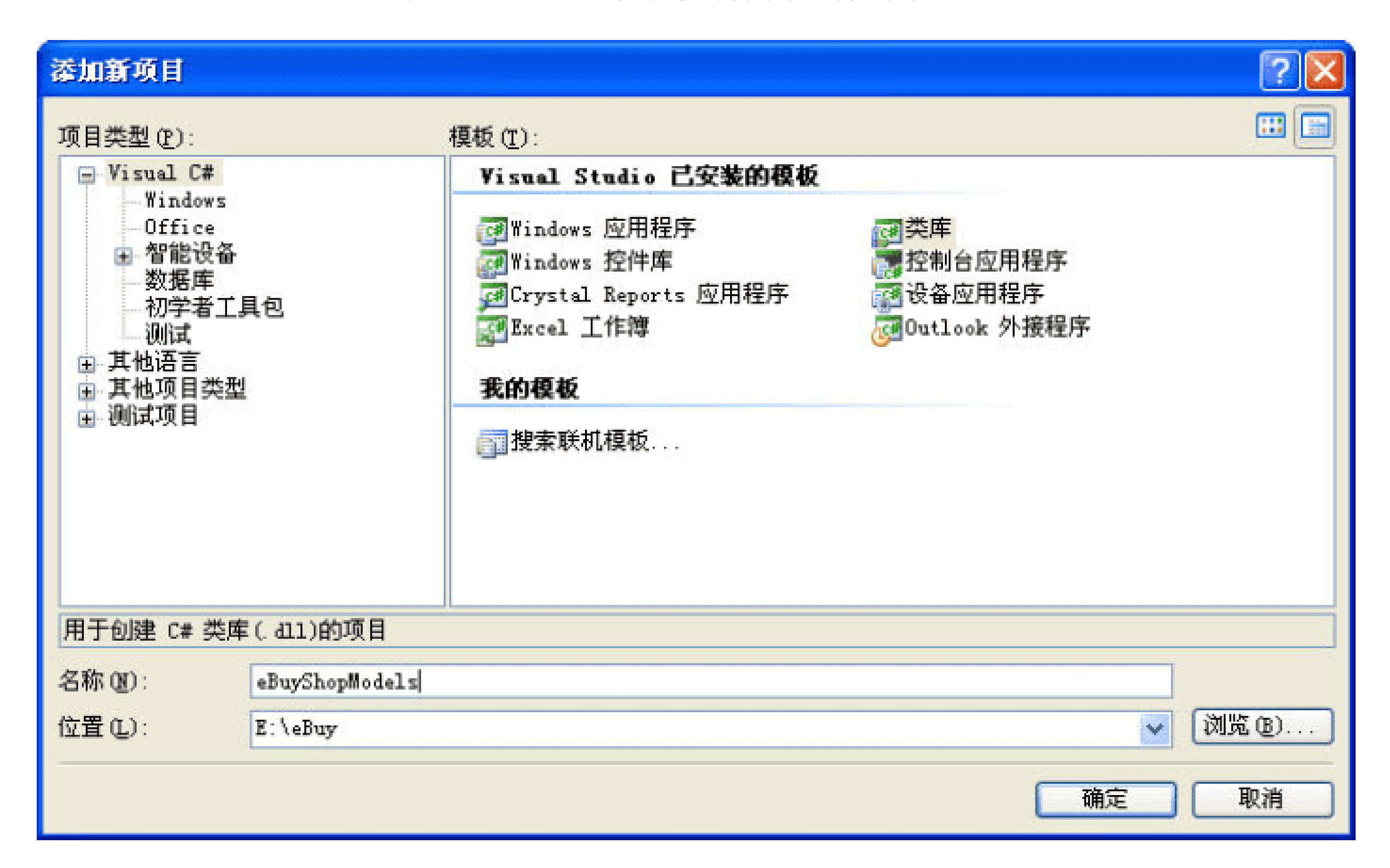

图 3-7 创建模型层项目

3.1.4 任务评价

标 准	掌 握 情 况
能使用 Visual Studio 建立项目	熟练☐ 基本☐ 模糊☐
能在已有工程中添加项目	熟练☐ 基本☐ 模糊☐
能理解三层开发的开发方式	熟练☐ 基本☐ 模糊☐
自 我 小 结	

3.1.5　任务拓展

读者可以按照本节的内容在学习开发的项目中自己创建这种三层开发的结构。另外读者在模仿开发本书项目的同时，可以另外再自行选题开发一个类似的练习项目，并将从本书中学习的知识用到自己的项目开发中，这样可以得到更好的学习效果。

任务 3.2　设计模型层

3.2.1　任务分析

任务目标：完成系统整体框架的搭建后，开始对每一层里面的内容进行独立开发。在这个任务中，将要设计用户管理模块的模型层，主要包括 User.cs、UserRole.cs 和 UserState.cs 3 个文件。

完成标准：能完成用户管理模块的模型层内容的设计。

应用手段：建立用户管理模块中的实体类。

3.2.2　相关知识

用户有状态和角色之分，这些内容在数据库中表现为外键关系。处理外键一般有两种方式：使用外键表 ID 或者使用外键对象。使用外键表 ID 的方法比较简单，但目前比较流行的是使用外键对象的方式，其好处是可以依据外键类直接访问外键的其他属性。

3.2.3　任务实施

1．创建 User 类

右击“解决方案资源管理器”中的“eBuyShopModels”，在弹出的快捷菜单中选择“添加”|“类（C）”命令，打开如图 3-8 所示的“添加新项”对话框。在“模板”列表中选择“类”，并将其名称改为“User.cs”，单击“确定”按钮完成 User 类的创建。

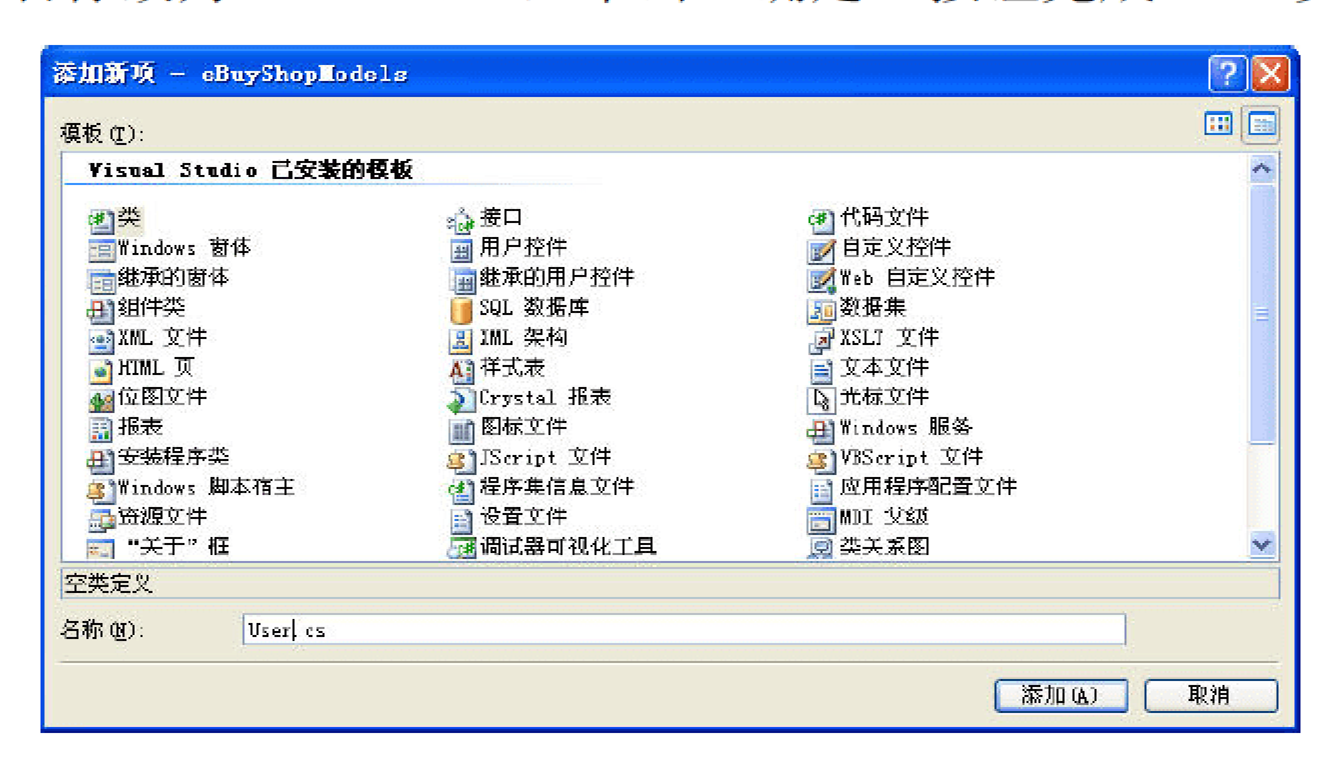

图 3-8　创建 User 类

User 类的主要代码如下：

```
using System;
using System.Collections.Generic;
using System.Text;
namespace eBuyShop.Models
{
   [Serializable()]
   public class User
      {
            private int id;
            private UserState userState;
            private UserRole userRole;
            private string loginId = String.Empty;
            private string loginPwd = String.Empty;
            private string name = String.Empty;
            private string address = String.Empty;
            private string phone = String.Empty;
            private string mail = String.Empty;
            public User() { }
            public int Id
            {
            get { return this.id; }
                  set { this.id = value; }
            }
            public UserState UserState
            {
            get { return this.userState; }
            set { this.userState = value; }
            }
            public UserRole UserRole
            {
            get { return this.userRole; }
                  set { this.userRole = value; }
            }
            public string LoginId
            {
                  get { return this.loginId; }
                  set { this.loginId = value; }
            }
            public string LoginPwd
            {
                  get { return this.loginPwd; }
                  set { this.loginPwd = value; }
            }
            public string Name
            {
                  get { return this.name; }
                  set { this.name = value; }
```

```
            }
            public string Address
            {
                get { return this.address; }
                set { this.address = value; }
            }
            public string Phone
            {
                get { return this.phone; }
                set { this.phone = value; }
            }
            public string Mail
            {
                get { return this.mail; }
                set { this.mail = value; }
            }
        }
}
```

2．创建 UserRole.cs 类

按照创建 User 类的方法，创建 UserRole.cs 类，用于表示不同的用户权限，其主要代码如下：

```
using System;
using System.Collections.Generic;
using System.Text;
namespace eBuyShop.Models
{
    [Serializable()]
    public class UserRole
        {
            private int id;
            private string name = String.Empty;
            public UserRole() { }
            public int Id
            {
                get { return this.id; }
                set { this.id = value; }
            }
            public string Name
            {
                get { return this.name; }
                set { this.name = value; }
            }
        }
}
```

3．创建 UserState.cs 类

按照创建 User 类的方法，创建 UserState.cs 类，用于表示用户的状态，其主要代码如下：

```
using System;
using System.Collections.Generic;
using System.Text;
namespace eBuyShop.Models
{
    [Serializable()]
    public class UserState
    {
        private int id;
        private string name = String.Empty;
        public UserState() { }
        public int Id
        {
            get { return this.id; }
            set { this.id = value; }
        }
        public string Name
        {
            get { return this.name; }
            set { this.name = value; }
        }
    }
}
```

对于模型层类的设计，本书不一一列举，读者可以参考本节中的内容自行将模型层中的其他实体类设计出来。在设计实体类时要注意，类的属性和方法的设定应满足前面需求分析中提到的功能要求。

3.2.4 任务评价

标　准	掌 握 情 况
能理解模型层在三层开发中的作用	熟练□ 基本□ 模糊□
能使用 Visual Studio 添加类	熟练□ 基本□ 模糊□
能根据系统的要求设计实体类	熟练□ 基本□ 模糊□
自 我 小 结	

3.2.5 任务拓展

本节完成了对 User 类相关的模型层内容的开发，读者可以参照本任务中的内容完成 Order.cs 等实体类的设计，并完成自己练习项目中相关内容的设计。

任务 3.3　设计数据访问层

3.3.1　任务分析

任务目标：数据访问层的主要作用是实现系统对数据库的各项操作，是真正实现了数据库操作的一层。本任务中，将针对模型层中创建的 3 个类，在数据访问层中创建 UserService.cs、UserRoleService.cs 以及 UserStateService.cs 3 个类，实现对应的实体数据处理。

完成标准：能针对前面模型层的内容完成数据访问层的设计。

应用手段：在类中实现对数据库的各种操作。

3.3.2　相关知识

数据访问层封装了所有与数据库交互的操作，如增、删、改以及查等，数据访问层可针对每个数据表提供增、删、改以及查操作，而不必作出业务逻辑上的判断。

3.3.2.1　数据访问层的命名

数据访问层项目一般命名为“DAL”，或者“解决方案名+DAL”。在本书的项目中，命名为 eBuyShopDAL，命名空间为 eBuyShop.DAL。

针对模型层中的每一个类，数据访问层有一个对应的数据访问类。例如，针对 User 实体类，有一个对应的 UserService 类，用以处理有关 Users 表的数据。

3.3.2.2　数据处理

1．增

一般增加方法处理较为简单，这里以用户为例，增加方法代码为：

```
public static User AddUser(User user)
```

使用静态方法的优点是使用方便，用的时候不用对数据类进行实例化。由于数据访问层类中一般只有方法，因此，可以直接将 UserService 类写成静态类。

2．删

典型的删除方法代码如下：

```
public static void DeleteUser(User user)
public static void DeleteUserById(int id)
```

3．改

修改方法代码如下：

```
public static void ModifyUser(User user)
```

4．查

典型的查询方法代码如下：

```
public static User GetUserById(int id)
public static User GetUserByLoginId(string loginId)
```

3.3.3 任务实施

1．创建 UserService 类

在数据访问层中创建一个名为“UserService”的类。添加之后，对 User 的操作都将放在这个类中。UserService.cs 类的主要代码如下：

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Data;
using System.Data.SqlClient;
using eBuyShop.Models;
namespace eBuyShop.DAL
{
    public static partial class UserService
    {
    #region 改
    /// <summary>
    /// 更改会员状态
    /// </summary>
    /// <param name="id"></param>
    public static void ModifyUserStatusById(int id)
    {
        int status=0;
        User user=GetUserById(id);
        if (user.UserState.Id == 1)
    {
        status = 2;
      }
      else
      {
            status = 1;
      }
             string sql = "Update users SET userstateid ="+ status +" WHERE Id = @UserId";
             DBHelper.ExecuteCommand(sql, new SqlParameter("@UserId", id));
        }
        /// <summary>
        /// 更改会员状态
        /// </summary>
        /// <param name="id"></param>
        /// <param name="status"></param>
        public static void ModifyUserStatus(int id,int status)
```

```
{
    string sql = "Update users SET userstateid =" + status + " WHERE Id = @UserId";

    DBHelper.ExecuteCommand(sql, new SqlParameter("@UserId", id));
}
public static void ModifyUser(User user)
{
    string sql =
        "UPDATE Users " +
        "SET " +
            "UserStateId = @UserStateId, " + //FK
            "UserRoleId = @UserRoleId, " + //FK
            "LoginId = @LoginId, " +
            "LoginPwd = @LoginPwd, " +
            "Name = @Name, " +
            "Address = @Address, " +
            "Phone = @Phone, " +
            "Mail = @Mail " +
        "WHERE Id = @Id";
    SqlParameter[] para = new SqlParameter[]
    {
        new SqlParameter("@Id", user.Id),
        new SqlParameter("@UserStateId", user.UserState.Id), //FK
        new SqlParameter("@UserRoleId", user.UserRole.Id), //FK
        new SqlParameter("@LoginId", user.LoginId),
        new SqlParameter("@LoginPwd", user.LoginPwd),
        new SqlParameter("@Name", user.Name),
        new SqlParameter("@Address", user.Address),
        new SqlParameter("@Phone", user.Phone),
        new SqlParameter("@Mail", user.Mail)
    };
    DBHelper.ExecuteCommand(sql, para);
}
#endregion
#region 查
/// <summary>
/// 查询所有普通用户
/// </summary>
/// <returns></returns>
public static IList<User> GetAllNormalUsers()
{
    string sql = "SELECT * FROM users WHERE userstateid = 1" ;
    return GetUsersBySql(sql);
}
/// <summary>
/// 通过登录名查找管理员
/// </summary>
/// <param name="loginId"></param>
/// <returns></returns>
```

```
        public static User GetAdminUserByLoginId(string loginId)
        {
            string sql = "SELECT * FROM users WHERE LoginId = @LoginId and UserRoleId=
@RoleId";
            int roleId;
            int userStateId;

            SqlParameter[] para = new SqlParameter[]
            {
                new SqlParameter("@LoginId", loginId),
                new SqlParameter("@RoleId", 3)
            };
            using (SqlDataReader reader = DBHelper.GetReader(sql, para))
            {
                if (reader.Read())
                {
                    User user = new User();
                    user.Id = (int)reader["Id"];
                    user.LoginId = (string)reader["LoginId"];
                    user.LoginPwd = (string)reader["LoginPwd"];
                    user.Name = (string)reader["Name"];
                    user.Address = (string)reader["Address"];
                    user.Phone = (string)reader["Phone"];
                    user.Mail = (string)reader["Mail"];
                    roleId = (int)reader["UserRoleId"];
                    userStateId = (int)reader["UserStateId"];
                    user.UserRole = UserRoleService.GetUserRoleById(roleId);
                    user.UserState = UserStateService.GetUserStateById(userStateId);
                    reader.Close();
                    return user;
                }
                else
                {
                    reader.Close();
                    return null;
                }
            }
        }
        public static IList<User> GetAllUsers()
        {
            string sqlAll = "SELECT * FROM Users";
            return GetUsersBySql(sqlAll);
        }
        /// <summary>
        ///  根据 id 查询单个用户
        /// </summary>
        /// <param name="id"></param>
        /// <returns></returns>
        public static User GetUserById(int id)
```

```
        {
            string sql = "SELECT * FROM Users WHERE Id = @Id";
            int userStateId;
            int userRoleId;
            using (SqlDataReader reader = DBHelper.GetReader(sql, new SqlParameter("@Id", id)))
            {
                if (reader.Read())
                {
                    User user = new User();
                    user.Id = (int)reader["Id"];
                    user.LoginId = (string)reader["LoginId"];
                    user.LoginPwd = (string)reader["LoginPwd"];
                    user.Name = (string)reader["Name"];
                    user.Address = (string)reader["Address"];
                    user.Phone = (string)reader["Phone"];
                    user.Mail = (string)reader["Mail"];
                    userStateId = (int)reader["UserStateId"]; //FK
                    userRoleId = (int)reader["UserRoleId"]; //FK
                    reader.Close();
                    user.UserState = UserStateService.GetUserStateById(userStateId);
                    user.UserRole = UserRoleService.GetUserRoleById(userRoleId);
                    return user;
                }
                else
                {
                    reader.Close();
                    return null;
                }
            }
        }
        /// <summary>
        /// 根据登录名查询用户
        /// </summary>
        /// <param name="loginId"></param>
        /// <returns></returns>
        public static User GetUserByLoginId(string loginId)
        {
            string sql = "SELECT * FROM Users WHERE LoginId = @LoginId";
            int userStateId;
            int userRoleId;
            using (SqlDataReader reader = DBHelper.GetReader(sql, new SqlParameter("@LoginId",
loginId)))
            {
                if (reader.Read())
                {
                    User user = new User();

                    user.Id = (int)reader["Id"];
                    user.LoginId = (string)reader["LoginId"];
```

```
                    user.LoginPwd = (string)reader["LoginPwd"];
                    user.Name = (string)reader["Name"];
                    user.Address = (string)reader["Address"];
                    user.Phone = (string)reader["Phone"];
                    user.Mail = (string)reader["Mail"];
                    userStateId = (int)reader["UserStateId"]; //FK
                    userRoleId = (int)reader["UserRoleId"]; //FK
                    reader.Close();
                    user.UserState = UserStateService.GetUserStateById(userStateId);
                    user.UserRole = UserRoleService.GetUserRoleById(userRoleId);

                    return user;
                }
                else
                {
                    reader.Close();
                    return null;
                }
            }
        }
        /// <summary>
        ///  依据 sql 语句查询用户
        /// </summary>
        /// <param name="safeSql"></param>
        /// <returns></returns>
        private static IList<User> GetUsersBySql(string safeSql)
        {
            List<User> list = new List<User>();

            using (DataTable table = DBHelper.GetDataSet(safeSql))
            {
                foreach (DataRow row in table.Rows)
                {
                    User user = new User();

                    user.Id = (int)row["Id"];
                    user.LoginId = (string)row["LoginId"];
                    user.LoginPwd = (string)row["LoginPwd"];
                    user.Name = (string)row["Name"];
                    user.Address = (string)row["Address"];
                    user.Phone = (string)row["Phone"];
                    user.Mail = (string)row["Mail"];
                    user.UserState = UserStateService.GetUserStateById((int)row["UserStateId"]); //FK
                    user.UserRole = UserRoleService.GetUserRoleById((int)row["UserRoleId"]);   //FK
                    list.Add(user);
                }
                return list;
            }
        }
```

```
/// <summary>
/// 根据 sql 及相关参数查询用户
/// </summary>
/// <param name="sql"></param>
/// <param name="values"></param>
/// <returns></returns>
private static IList<User> GetUsersBySql(string sql, params SqlParameter[] values)
{
    List<User> list = new List<User>();

    using (DataTable table = DBHelper.GetDataSet(sql, values))
    {
        foreach (DataRow row in table.Rows)
        {
            User user = new User();

            user.Id = (int)row["Id"];
            user.LoginId = (string)row["LoginId"];
            user.LoginPwd = (string)row["LoginPwd"];
            user.Name = (string)row["Name"];
            user.Address = (string)row["Address"];
            user.Phone = (string)row["Phone"];
            user.Mail = (string)row["Mail"];
            user.UserState = UserStateService.GetUserStateById((int)row["UserStateId"]); //FK
            user.UserRole = UserRoleService.GetUserRoleById((int)row["UserRoleId"]);  //FK
            list.Add(user);
        }
        return list;
    }
}
#endregion
#region 增
/// <summary>
/// 添加新用户
/// </summary>
/// <param name="user"></param>
/// <returns></returns>
public static User AddUser(User user)
{
    string sql =
        "INSERT Users (LoginId, LoginPwd, Name, Address, Phone, Mail, UserRoleId,
UserStateId)" +
        "VALUES (@LoginId, @LoginPwd, @Name, @Address, @Phone, @Mail,
@UserRoleId, @UserStateId)";
    sql += " ; SELECT @@IDENTITY";
    SqlParameter[] para = new SqlParameter[]
    {
        new SqlParameter("@UserStateId", user.UserState.Id),    //FK
        new SqlParameter("@UserRoleId", user.UserRole.Id),      //FK
        new SqlParameter("@LoginId", user.LoginId),
```

```
                new SqlParameter("@LoginPwd", user.LoginPwd),
                new SqlParameter("@Name", user.Name),
                new SqlParameter("@Address", user.Address),
                new SqlParameter("@Phone", user.Phone),
                new SqlParameter("@Mail", user.Mail)
            };
            int newId = DBHelper.GetScalar(sql, para);
            return GetUserById(newId);
        }
        #endregion
        #region  删
        /// <summary>
        ///  删除用户
        /// </summary>
        /// <param name="user"></param>
        public static void DeleteUser(User user)
        {
            DeleteUserById(user.Id);
        }
        /// <summary>
        ///  根据 id 删除用户
        /// </summary>
        /// <param name="id"></param>
        public static void DeleteUserById(int id)
        {
            string sql = @"DELETE Orders where UserId=@Id
            DELETE Users WHERE Id = @Id";
            SqlParameter[] para = new SqlParameter[]
            {
                new SqlParameter("@Id", id)
            };
            DBHelper.ExecuteCommand(sql, para);
        }
        #endregion
    }
}
```

知识点小贴士

通过阅读上面的代码，读者应该会发现，UserService 类对 User 的增、删、改以及查都各自提供了几种不同方法。为什么对一种操作要给出不同的实现方法呢？这是由系统的功能来决定的。例如，查询功能，有时候需要按照商品的编号来进行查询，而有些时候则需要根据商品的名称来进行查询，因此在设计时需要提供不同的实现方法。当然，在方法设计的过程中，一开始也不可能将所有方法都考虑进来，可以在后面的开发中根据具体需要往类里面添加新的方法，以满足新的功能需求。

2．创建 UserRoleService 类

采用类似的方式，在数据访问层中添加 UserRoleService 类，与实体类 UserRole 相关的

操作都将保存在这个类中。UserRoleService.cs 类的主要代码如下：

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Data;
using System.Data.SqlClient;
using eBuyShop.Models;
namespace eBuyShop.DAL
{
    public static partial class UserRoleService
    {
        #region 增
        public static UserRole AddUserRole(UserRole userRole)
        {
            string sql ="INSERT UserRoles (Name)" +"VALUES (@Name)";
            sql += " ; SELECT @@IDENTITY";
            try
            {
                SqlParameter[] para = new SqlParameter[]
                {
                    new SqlParameter("@Name", userRole.Name)
                };
                int newId = DBHelper.GetScalar(sql, para);
                return GetUserRoleById(newId);
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
                return null;
            }
        }
        #endregion
        #region 删
        public static void DeleteUserRole(UserRole userRole)
        {
            DeleteUserRoleById( userRole.Id );
        }
        public static void DeleteUserRoleById(int id)
        {
            string sql = "DELETE UserRoles WHERE Id = @Id";
            try
            {
                SqlParameter[] para = new SqlParameter[]
                {
                    new SqlParameter("@Id", id)
                };
                DBHelper.ExecuteCommand(sql, para);
            }
```

```
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
            throw e;
        }
    }
    public static void DeleteUserRoleByName(string name)
    {
        string sql = "DELETE UserRoles WHERE Name = @Name";
        try
        {
            SqlParameter[] para = new SqlParameter[]
            {
                new SqlParameter("@Name", name)
            };
            DBHelper.ExecuteCommand(sql, para);
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
            throw e;
        }
    }
    #endregion
    #region 改
    public static void ModifyUserRole(UserRole userRole)
    {
        string sql ="UPDATE UserRoles " +"SET " +"Name = @Name " +
                  "WHERE Id = @Id";
        try
        {
            SqlParameter[] para = new SqlParameter[]
            {
                new SqlParameter("@Id", userRole.Id),
                new SqlParameter("@Name", userRole.Name)
            };
            DBHelper.ExecuteCommand(sql, para);
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
            throw e;
        }
    }
    #endregion
    #region 查
    public static IList<UserRole> GetAllUserRoles()
    {
        string sqlAll = "SELECT * FROM UserRoles";
```

```
        return GetUserRolesBySql( sqlAll );
    }
    public static UserRole GetUserRoleById(int id)
    {
        string sql = "SELECT * FROM UserRoles WHERE Id = @Id";
        try
        {
            SqlDataReader reader = DBHelper.GetReader(sql, new SqlParameter("@Id", id));
            if (reader.Read())
            {
                UserRole userRole = new UserRole();
                userRole.Id = (int)reader["Id"];
                userRole.Name = (string)reader["Name"];
                reader.Close();
                return userRole;
            }
            else
            {
                reader.Close();
                return null;
            }
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
            return null;
        }
    }
    public static UserRole GetUserRoleByName(string name)
    {
        string sql = "SELECT * FROM UserRoles WHERE Name = @Name";
        try
        {
            SqlDataReader reader = DBHelper.GetReader(sql, new SqlParameter("@Name", name));
            if (reader.Read())
            {
                UserRole userRole = new UserRole();
                userRole.Id = (int)reader["Id"];
                userRole.Name = (string)reader["Name"];
                reader.Close();
                return userRole;
            }
            else
            {
                reader.Close();
                return null;
            }
        }
        catch (Exception e)
```

```
                {
                    Console.WriteLine(e.Message);
                    return null;
                }
            }
            private static IList<UserRole> GetUserRolesBySql( string safeSql )
            {
                List<UserRole> list = new List<UserRole>();
                try
                {
                    DataTable table = DBHelper.GetDataSet( safeSql );
                    foreach (DataRow row in table.Rows)
                    {
                        UserRole userRole = new UserRole();
                        userRole.Id = (int)row["Id"];
                        userRole.Name = (string)row["Name"];
                        list.Add(userRole);
                    }
                    return list;
                }
                catch (Exception e)
                {
                    Console.WriteLine(e.Message);
                    return null;
                }
            }
            private static IList<UserRole> GetUserRolesBySql( string sql, params SqlParameter[] values )
            {
                List<UserRole> list = new List<UserRole>();
                try
                {
                    DataTable table = DBHelper.GetDataSet( sql, values );
                    foreach (DataRow row in table.Rows)
                    {
                        UserRole userRole = new UserRole();
                        userRole.Id = (int)row["Id"];
                        userRole.Name = (string)row["Name"];
                        list.Add(userRole);
                    }
                    return list;
                }
                catch (Exception e)
                {
                    Console.WriteLine(e.Message);
                    return null;
                }
            }
            #endregion
        }
    }
```

知识点小贴士

比较前面创建的两个类，可以看到它们在方法命名上采用了同样的方式，都包含了“By”以及“All”等关键词。关键词“By”表示后面将按什么条件来进行操作，而包含“All”的方法则表示针对所有的记录，如果没有“All”一般都是针对一条记录或某些记录。当然，这种命名的方式不是绝对的，但由于数据访问层以及逻辑层中的方法很多，因此在命名时最好采用同一套规范，以方便后期的开发和维护。

3．创建 UserStateService 类

同样，在数据访问层中添加 UserStateService 类。添加之后，对 UserState 实体类进行的操作都将保存在这个类中。UserStateService.cs 类的主要代码如下：

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Data;
using System.Data.SqlClient;
using eBuyShop.Models;
namespace eBuyShop.DAL
{
    public static partial class UserStateService
    {
        /// <summary>
        ///  取得默认状态
        /// </summary>
        /// <returns></returns>
        public static UserState GetNormalUserState()
        {
            return GetUserStateById(1);
        }
        #region  增
        public static UserState AddUserState(UserState userState)
        {
            string sql =
                "INSERT UserStates (Name)" +
                "VALUES (@Name)";
            sql += " ; SELECT @@IDENTITY";
            try
            {
                SqlParameter[] para = new SqlParameter[]
                {
                    new SqlParameter("@Name", userState.Name)
                };
                int newId = DBHelper.GetScalar(sql, para);
                return GetUserStateById(newId);
            }
            catch (Exception e)
            {
```

```
                Console.WriteLine(e.Message);
                return null;
            }
        }
        #endregion
        #region 删
        public static void DeleteUserState(UserState userState)
        {
            DeleteUserStateById(userState.Id);
        }
        public static void DeleteUserStateById(int id)
        {
            string sql = "DELETE UserStates WHERE Id = @Id";
            try
            {
                SqlParameter[] para = new SqlParameter[]
                {
                    new SqlParameter("@Id", id)
                };
                DBHelper.ExecuteCommand(sql, para);
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
                throw e;
            }
        }
        public static void DeleteUserStateByName(string name)
        {
            string sql = "DELETE UserStates WHERE Name = @Name";
            try
            {
                SqlParameter[] para = new SqlParameter[]
                {
                    new SqlParameter("@Name", name)
                };
                DBHelper.ExecuteCommand(sql, para);
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
                throw e;
            }
        }
        #endregion
        #region 改
        public static void ModifyUserState(UserState userState)
        {
            string sql =
```

```
            "UPDATE UserStates " +
            "SET " +
            "Name = @Name " +
            "WHERE Id = @Id";
        try
        {
            SqlParameter[] para = new SqlParameter[]
            {
                new SqlParameter("@Id", userState.Id),
                new SqlParameter("@Name", userState.Name)
            };
            DBHelper.ExecuteCommand(sql, para);
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
            throw e;
        }
    }
    #endregion
    #region 查
    public static IList<UserState> GetAllUserStates()
    {
        string sqlAll = "SELECT * FROM UserStates";
        return GetUserStatesBySql(sqlAll);
    }
    public static UserState GetUserStateById(int id)
    {
        string sql = "SELECT * FROM UserStates WHERE Id = @Id";
        try
        {
            SqlDataReader reader = DBHelper.GetReader(sql, new SqlParameter("@Id", id));
            if (reader.Read())
            {
                UserState userState = new UserState();
                userState.Id = (int)reader["Id"];
                userState.Name = (string)reader["Name"];
                reader.Close();
                return userState;
            }
            else
            {
                reader.Close();
                return null;
            }
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
```

```
                return null;
            }
        }
        public static UserState GetUserStateByName(string name)
        {
            string sql = "SELECT * FROM UserStates WHERE Name = @Name";
            try
            {
                SqlDataReader reader = DBHelper.GetReader(sql, new SqlParameter("@Name", name));
                if (reader.Read())
                {
                    UserState userState = new UserState();
                    userState.Id = (int)reader["Id"];
                    userState.Name = (string)reader["Name"];
                    reader.Close();
                    return userState;
                }
                else
                {
                    reader.Close();
                    return null;
                }
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
                return null;
            }
        }
        private static IList<UserState> GetUserStatesBySql(string safeSql)
        {
            List<UserState> list = new List<UserState>();
            try
            {
                DataTable table = DBHelper.GetDataSet(safeSql);
                foreach (DataRow row in table.Rows)
                {
                    UserState userState = new UserState();
                    userState.Id = (int)row["Id"];
                    userState.Name = (string)row["Name"];
                    list.Add(userState);
                }
                return list;
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
                return null;
            }
        }
```

```
        private static IList<UserState> GetUserStatesBySql(string sql, params SqlParameter[] values)
        {
            List<UserState> list = new List<UserState>();
            try
            {
                DataTable table = DBHelper.GetDataSet(sql, values);
                foreach (DataRow row in table.Rows)
                {
                    UserState userState = new UserState();
                    userState.Id = (int)row["Id"];
                    userState.Name = (string)row["Name"];
                    list.Add(userState);
                }
                return list;
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
                return null;
            }
        }
        #endregion
    }
}
```

通过以上代码，在数据访问层中实现了与 User 相关的 3 个实体类的数据操作。从中可以看到，数据访问层的设计其实就是数据库操作方法的设计。由于功能的需要，对于每一种操作都需要提供几种不同的操作方法。本节中 3 个数据访问类中包含了在数据层开发时通常会采用的一些方法定义，读者可以参照这 3 个类完成项目中其他数据层访问类的设计。

3.3.4　任务评价

标　　准	掌 握 情 况
能理解数据访问层在三层开发中的作用	熟练□　基本□　模糊□
能在数据访问层中实现“增”	熟练□　基本□　模糊□
能在数据访问层中实现“删”	熟练□　基本□　模糊□
能在数据访问层中实现“改”	熟练□　基本□　模糊□
能在数据访问层中实现“查”	熟练□　基本□　模糊□
自 我 小 结	

3.3.5 任务拓展

读者可以参照本节中的内容完成数据访问层中用于订单处理的 OrderService.cs 类等数据访问层类的设计，并完成自己练习项目中相关内容的设计。

任务 3.4 设计业务逻辑层

3.4.1 任务分析

任务目标：业务逻辑层是连接表示层与数据访问层、并封装了符合系统功能要求的部分。本任务中，将完成业务逻辑层 3 个类的编写，它们分别是 UserManager.cs、UserRoleManager.cs 和 UserStateManager.cs。

完成标准：完成业务逻辑层中相关类的编写。

应用手段：按系统功能需求编写逻辑层中类的方法。

3.4.2 相关知识

业务逻辑层是表示层和数据访问层的桥梁，负责业务处理和数据传递。该部分的方法一般与实际需求相关，主要用于做一些有效性验证的工作，更好地保证程序运行的顺畅性。例如，完成数据添加、修改和查询业务等；不允许指定的文本框中输入空字符串，数据格式是否正确以及数据类型验证；用户权限的合法性判断等。通过这些判断来决定是否将操作继续向后传递，从而保证程序的正常运行。

3.4.3 任务实施

1．创建 UserManager 类

在逻辑层中新建 UserManager 类。添加之后，表示层中对 User 进行的操作都将通过调用该 UserManager 类中的方法来实现。UserManager.cs 类的主要代码如下：

```
using System;
using System.Collections.Generic;
using System.Text;
using eBuyShop.DAL;
using eBuyShop.Models;
namespace eBuyShop.BLL
{
    public static partial class UserManager
    {
        /// <summary>
        ///  登录
```

```
/// </summary>
/// <param name="loginId">登录名</param>
/// <param name="loginPwd">登录密码</param>
/// <param name="validUser">输出用户</param>
/// <returns></returns>
public static bool Login(string loginId, string loginPwd, out User validUser)
{
    User user = UserService.GetUserByLoginId(loginId);
    if (user == null)
    {
        //用户名不存在
        validUser = null;
        return false;
    }
    if (user.LoginPwd == loginPwd)
    {
        validUser = user;
        return true;
    }
    else
    {
        //密码错误
        validUser = null;
        return false;
    }
}
/// <summary>
/// 注册新用户
/// </summary>
/// <param name="user"></param>
/// <returns></returns>
public static bool Register(User user)
{
    if (LoginIdExists(user.LoginId))
    {
        return false;
    }
    else
    {
        AddUser(user);
        return true;
    }
}
/// <summary>
/// 管理员登录
/// </summary>
/// <param name="loginId"></param>
/// <param name="loginPwd"></param>
/// <param name="validUser"></param>
```

```
/// <returns></returns>
public static bool AdminLogin(string loginId, string loginPwd, out User validUser)
{
    User user=UserService.GetAdminUserByLoginId(loginId);
    if (user == null)
    {
        //用户名不存在
        validUser = null;
        return false;
    }
    if (user.LoginPwd == loginPwd)
    {
        validUser = user;
        return true;
    }
    else
    {
        //密码错误
        validUser = null;
        return true;
    }
}
/// <summary>
///
/// </summary>
/// <param name="loginId"></param>
/// <returns></returns>
public static bool LoginIdExists(string loginId)
{
    if (UserService.GetUserByLoginId(loginId) == null)
        return false;
    return true;
}
#region 改
/// <summary>
/// 修改基本信息
/// </summary>
/// <param name="Name"></param>
/// <param name="Phone"></param>
/// <param name="Address"></param>
/// <param name="Mail"></param>
/// <param name="Id"></param>
public static void ModifyBasicInfo( string Name, string Phone, string Address, string
Mail,int Id)
{
    User user = UserService.GetUserById(Id);
    user.Name = Name;
    user.Phone = Phone;
    user.Address = Address;
```

```
        user.Mail = Mail;
        UserService.ModifyUser(user);
    }
    /// <summary>
    /// 根据 id 修改用户状态
    /// </summary>
    /// <param name="userId"></param>
    public static void ModifyUserStatusById(int userId)
    {
        UserService.ModifyUserStatusById(userId);
    }
    /// <summary>
    /// 修改用户状态
    /// </summary>
    /// <param name="userId"></param>
    /// <param name="status"></param>
    public static void ModifyUserStatus(int userId, int status)
    {
        UserService.ModifyUserStatus(userId, status);
    }
    public static void ModifyUser(User user)
    {
        UserService.ModifyUser(user);
    }
    #endregion
    #region 查
    /// <summary>
    /// 获得所有普通用户
    /// </summary>
    /// <returns></returns>
    public static IList<User> GetAllNormalUsers()
    {
        return UserService.GetAllNormalUsers();
    }
    public static IList<User> GetAllUsers()
    {
        return UserService.GetAllUsers();
    }
    public static User GetUserById(int id)
    {
        return UserService.GetUserById(id);
    }
    #endregion
    #region 增
    public static User AddUser(User user)
    {
        if (user.UserState == null)
        {
            user.UserState = UserStateManager.GetDefaultUserState();
```

```
                }
                if (user.UserRole == null)
                {
                    user.UserRole = UserRoleManager.GetDefaultUserRole();
                }
                return UserService.AddUser(user);
            }
            #endregion
            #region 删
            public static void DeleteUser(User user)
            {
                UserService.DeleteUser(user);
            }
            public static void DeleteUserById(int id)
            {
                UserService.DeleteUserById(id);
            }
            #endregion
        }
}
```

知识点小贴士

从以上代码可以发现，逻辑层中的方法没有直接参与数据库的操作，而是通过调用数据访问层中的方法来实现。逻辑层将数据访问层中的数据库操作重新组合在一起以实现更为复杂的逻辑功能。逻辑层中有一些方法，直接调用数据访问层中的对应方法就可以实现，比如上述代码中的 DeleteUserById 方法。逻辑层中还有一些方法，由于涉及的逻辑比较复杂，因此需要调用数据访问层的若干方法才能够实现，如上述代码中的 AddUser 方法。可以说，逻辑层中所涉及的功能是对需求分析最直接的体现，同时，由于逻辑层封装了系统的主要逻辑，因此也是一个系统的真正价值所在。

2．创建 UserRoleManager 类

在逻辑层中新建 UserRoleManager 类。添加之后，表示层中对 UserRole 进行的操作都将通过调用该类中的方法来实现。UserRoleManager.cs 类的主要代码如下：

```
using System;
using System.Collections.Generic;
using System.Text;
using eBuyShop.DAL;
using eBuyShop.Models;
namespace eBuyShop.BLL
{
    public static partial class UserRoleManager
    {
        #region 增
        public static UserRole AddUserRole(UserRole userRole)
        {
            return UserRoleService.AddUserRole(userRole);
```

```
            }
            #endregion
            #region 删
            public static void DeleteUserRole(UserRole userRole)
            {
                UserRoleService.DeleteUserRole(userRole);
            }
            public static void DeleteUserRoleById(int id)
            {
                UserRoleService.DeleteUserRoleById(id);
            }
            #endregion
            #region 改
            public static void ModifyUserRole(UserRole userRole)
            {
                UserRoleService.ModifyUserRole(userRole);
            }
            #endregion
            #region 查
            public static IList<UserRole> GetAllUserRoles()
            {
                return UserRoleService.GetAllUserRoles();
            }
            public static UserRole GetUserRoleById(int id)
            {
                return UserRoleService.GetUserRoleById(id);
            }
            public static UserRole GetDefaultUserRole()
            {
                return GetUserRoleById(1);
            }
            #endregion
    }
}
```

由于对 UserRole 进行操作所涉及的逻辑比较简单，因此其逻辑层中所涉及的方法也相对较少，基本上都是直接调用数据访问层中的对应方法来实现的。

3．创建 UserStateManager 类

同样，在逻辑层中新建一个 UserStateManager 类，添加之后，表示层中对 UserState 进行的操作都将通过调用该类中的方法来实现。UserStateManager.cs 类的主要代码如下：

```
using System;
using System.Collections.Generic;
using System.Text;
using eBuyShop.DAL;
using eBuyShop.Models;
namespace eBuyShop.BLL
{
    public static partial class UserStateManager
```

```
    {
        #region 增
        public static UserState AddUserState(UserState userState)
        {
            return UserStateService.AddUserState(userState);
        }
        #endregion
        #region 删
        public static void DeleteUserState(UserState userState)
        {
            UserStateService.DeleteUserState(userState);
        }
        public static void DeleteUserStateById(int id)
        {
            UserStateService.DeleteUserStateById(id);
        }
        #endregion
        #region 改
        public static void ModifyUserState(UserState userState)
        {
            UserStateService.ModifyUserState(userState);
        }
        #endregion
        #region 查
        public static IList<UserState> GetAllUserStates()
        {
            return UserStateService.GetAllUserStates();
        }
        public static UserState GetUserStateById(int id)
        {
            return UserStateService.GetUserStateById(id);
        }
        public static UserState GetDefaultUserState()
        {
            return GetUserStateById(1);
        }
        #endregion
    }
}
```

3.4.4 任务评价

标　　准	掌 握 情 况
能理解逻辑层在三层开发中的作用	熟练□ 基本□ 模糊□
能完成业务逻辑层中的“增”	熟练□ 基本□ 模糊□
能完成业务逻辑层中的“删”	熟练□ 基本□ 模糊□
能完成业务逻辑层中的“改”	熟练□ 基本□ 模糊□
能完成业务逻辑层中的“查”	熟练□ 基本□ 模糊□

续表

自 我 小 结

3.4.5　任务拓展

读者可以参照本节中的内容完成业务逻辑层中用于订单处理的 OrderManager.cs 类等数据访问层类的设计，完成自己练习项目中相关内容的设计。

任务 3.5　设计表示层

3.5.1　任务分析

任务目标：有了前面数据访问层和业务逻辑层的开发基础，现在可以加上表示层的用户接口来实现系统的功能模块。本任务中，主要实现用户登录界面的设计，以及通过调用相应业务逻辑层的方法实现对应的功能。

完成标准：完成用户登录界面的设计，并能够与业务逻辑层交换数据。

应用手段：设计页面并编写控件方法的实现与业务逻辑层的交互。

3.5.2　相关知识

ASP.NET 中的表示层负责内容的展现和与用户的交互，它给予用户最直接的体验。在许多项目中，表示层往往是项目成败的关键，而这点却常常被程序员忽视。

在 ASP.NET 中，表示层就是整个 Web 站点。具体的内容要根据需求分析来决定。例如，一个网络书店，有自己的用户系统，自然就需要有相关的用户登录、注册、管理等页面；图书系统，也就要有图书管理、图书列表、图书详细展示等页面；要实现在线销售，还需要有购物车、订单管理等页面。

对于 ASP.NET 程序员而言，表示层的主要内容就是控件+事件。如果仅仅是展示，可能只需要将控件绑定数据即可，不需要编写代码；但如果需要和用户交互，就要编写相关的事件代码。例如，在用户登录页面上，用户单击“登录”按钮便触发了登录事件，可能需要编写代码验证用户的输入内容是否合法，然后通过调用业务逻辑层的相关方法判断用户名和密码是否匹配。

3.5.3 任务实施

1．界面设计

在表示层中新建一个用户后台管理界面，包含两个文本框和两个按钮。两个文本框分别用来输入账号和密码，而两个按钮则分别用来进行输入数据提交和重置。当然，还需要对操作界面做一定的美工设计，不能直接将一个光秃秃的页面呈现给用户。界面大致的设计效果如图 3-9 所示。

图 3-9　用户后台管理登录界面

2．用户验证的实现

界面设计好后，接下来需要实现界面的功能。实现功能之前，首先需要了解操作的流程（与前面的功能设计项对应）。例如，在当前这个登录界面中，其用户操作的正常流程如下：输入账号→输入密码→单击“确定”按钮。当然，用户也可能由于某些原因在没有完整输入账号和密码的情况下单击了“确定”按钮，而这些情况也是在开发时需要考虑到的问题。既然主要的操作都由“确定”按钮来实现，那么就对这个按钮进行开发。在“确定”按钮的单击事件中，编写如下的验证方法：

```
protected void imgb_Sure_Click(object sender, ImageClickEventArgs e)
{
        User user;
        if (UserManager.AdminLogin(this.txtLoginId.Text, this.txtLoginPwd.Text, out user))
        {
            string strRedirect;
            //表单验证所指定的路径
            strRedirect = Request["ReturnUrl"];
            //表单验证
            System.Web.Security.FormsAuthentication.SetAuthCookie(user.Name, true);
            if(strRedirect==null)
                Response.Redirect("~/Admin/ListAllUsers.aspx");
            Response.Redirect(strRedirect);
        }
        else
        {
            Response.Redirect("~/Error.aspx");
        }
}
```

从上面的代码中可以看到，在验证方法中，并没有出现具体的用户验证，而是通过调用业务逻辑层中的 UserManager.AdminLogin 方法来实现的。而按钮单击事件中的方法则主要是完成对验证结果的处理，如果验证成功则跳转到 ListAllUsers.aspx 页面，若验证失败则跳转到 Error.aspx 页面。总之，在表示层中处理的主要是与用户交互相关的界面操作，而相应的功能实现则需要通过调用业务逻辑层中的内容来实现。

3.5.4　任务评价

标　　准	掌 握 情 况
能理解表示层在三层开发中的作用	熟练□　基本□　模糊□
能按系统要求进行表示层页面的设计	熟练□　基本□　模糊□
能实现表示层与业务逻辑层的交互	熟练□　基本□　模糊□
自 我 小 结	

3.5.5　任务拓展

读者可以参照本节的设计流程，自行设计用户注册页面，并通过调用对应的业务逻辑层中的方法来实现用户注册的功能。

本章小结

在建立的项目中利用了三层结构，对项目进行了整体的框架搭建，并且通过用户管理模块，演示了三层结构的开发方式。本章涉及的主要内容包括：三层结构（表示层、业务逻辑层和数据访问层）、进行三层结构开发的一般步骤、模型层的作用及开发方法、数据访问层的作用及开发方法、业务逻辑层的作用及开发方法、表示层的作用及开发方法以及各层之间的数据交互方式。

自我检测

一、选择题

1．假设网站需要增加一个新的功能，如对某类商品进行打折，则相关的代码应该写在（　）。

A．模型层　　　　B．数据访问层

C．业务逻辑层　　D．表示层

2．假设开发了一类新产品，为此单独创建了一个数据表，则与该数据表对应的实体类

应写在（ ）。

A．模型层 B．数据访问层

C．业务逻辑层 D．表示层

3．用户注册的方法，应该写在（ ）。

A．模型层 B．数据访问层

C．业务逻辑层 D．表示层

4．当需要添加一条查询语句时，应该添加在（ ）。

A．模型层 B．数据访问层

C．业务逻辑层 D．表示层

5．关于三层结构，不正确的是（ ）。

A．三层结构必须要模型层。

B．数据访问层必须要添加模型层的引用。

C．业务逻辑层必须要添加数据访问层的引用。

D．表示层必须要添加模型层、数据访问层、业务逻辑层的引用。

6．The following statements on 3-Tier Architecture, which is wrong（ ）.

A．The data access layer must add the reference of model layer.

B．The presentation layer must add the reference of model layer, data access layer and business logic layer.

C．The model layer is necessary in 3-tier architecture.

D．The business logic layer must add the reference of data access layer.

二、简答题

1．简述三层结构的优点。

2．简述数据访问层是如何完成增、删、改以及查操作的。

第 4 章　搭建页面框架

技能目标

1．能使用站点导航控件来实现站点导航。
2．能编辑站点地图文件进行站点导航控制。
3．能使用母版页来控制页面整体风格布局。
4．能使用现有静态页来制作母版页。

相关词汇

英 文 单 词	中 文 含 义	英 文 单 词	中 文 含 义
master		static	
sitemap		generate	
navigate		layout	
template		suffix	

工作任务

任务 4.1　设计母版页
任务 4.2　设计站点导航

任务 4.1　设计母版页

4.1.1　任务分析

任务目标：在这个任务中，将要进行母版页的设计，其主要目的是搭好页面框架，为后面的页面设计做准备，提高页面设计的效率并统一设计风格。本任务中设计的母版页用于后台管理程序母版，后台管理程序所涉及的页面都将由这个母版页来生成。这样，一方面提高了页面设计的效率，另一方面也保证了后台管理程序中页面的一致性。

完成标准：能完成后台管理页面母板页的设计。

应用手段：使用 Visual Studio 2005 中提供的母版页来进行实现。

4.1.2　相关知识

4.1.2.1　母版页

使用 ASP.NET 母版页可以为应用程序中的页创建一致的布局。单个母版页可以为应用程序中的所有页（或一组页）定义所需的外观和标准行为，然后可以创建包含要显示的内容的各个内容页。当用户请求内容页时，这些内容页将与母版页合并，从而输出母版页的布局与内容页中的内容组合。

母版页由两部分组成，即母版页本身与一个或多个内容页。

母版页为具有扩展名.master（如 MySite.master）的 ASP.NET 文件，它具有包括静态文本、HTML 元素和服务器控件的预定义布局。母版页由特殊的@Master 指令标识，该指令替换了用于普通.aspx 页的@Page 指令。

除@Master 指令外，母版页还包含页的所有顶级 HTML 元素，如 html、head 和 form。例如，在母版页上可以将一个 HTML 表用于布局、将一个 img 元素用于公司徽标、将静态文本用于版权声明并使用服务器控件创建站点的标准导航。在母版页中可以使用任何 HTML 元素和 ASP.NET 元素。

母版页的优点如下：

（1）母版页提供了开发人员已通过传统方式创建的功能，这些传统方式包括重复复制现有代码、文本和控件元素；使用框架集；对通用元素使用包含文件；使用 ASP.NET 用户控件等。

（2）使用母版页可以集中处理页的通用功能，只需在一个位置上进行更新即可。

（3）使用母版页可以方便地创建一组控件和代码，并将结果应用于一组页。例如，可以在母版页上使用控件来创建一个应用于所有页的菜单。

（4）母版页通过允许控制占位符控件的呈现方式，使开发者可以在细节上控制最终页的布局。

（5）母版页提供一个对象模型，使用该对象模型可以从各个内容页中自定义母版页。

页面运行时，母版页是按照下面的步骤处理的：

（1）用户通过输入内容页的 URL 来请求某页。

（2）获取该页后，读取@Page 指令。若该指令引用一个母版页，则也读取该母版页。若这是第一次请求这两个页，则两个页都要进行编译。

（3）包含更新的内容的母版页合并到内容页的控件树中。

（4）各个 Content 控件的内容合并到母版页中相应的 ContentPlaceHolder 控件中。

（5）浏览器中呈现得到的合并页。

4.1.2.2　ContentPlaceHolder 控件

ContentPlaceHolder 控件在母版页中定义相对内容区域，并呈现在内容页中找到的相关

Content 控件的所有文本、标记和服务器控件。

Content 控件使用其 ContentPlaceHolderID 属性与 ContentPlaceHolder 关联。将 ContentPlaceHolderID 属性设置为母版页中相关的 ContentPlaceHolder 控件的 ID 属性的值。在母版页中可以声明多个 ContentPlaceHolder 控件。

在内容页中，只有一个 Content 控件可以为母版页中的 ContentPlaceHolder 提供内容。但是，在使用某一母版页的每个内容页中，可以有不同的与 ContentPlaceHolder 相关联的 Content 控件。例如，可以在母版页中为页标题定义 ContentPlaceHolder，使用该母版页的每个内容页时，可以添加 Content 控件，该控件为页标题提供文本和标记。

4.1.3　任务实施

1．创建母版页

首先在表示层网站下新建一个名为“Admin”的文件夹，用于存放管理程序的页面，接下来右击 Admin 文件夹图标，在弹出的快捷菜单中选择“添加新项”|“母版页”命令，打开如图 4-1 所示的对话框。选择“母版页”选项，并将其名称改为“admin.master”，最后单击“添加”按钮完成母版页的创建。

图 4-1　母版页的创建

新创建的母版页上面只有一个 ContentPlaceHolder 控件，它是预留给内容页显示的控件，在实际的开发过程中，美工会制作静态页面，然后将美工制作的页面代码贴入母版页即可，注意保留一个 ContentPlaceHolder 控件，以后页面上变化的内容只是在 ContentPlaceHolder 控件里面呈现。

2．编辑母版页

根据界面设计的需要，对母版页进行了美化，并划分区域结构，其大致效果如图 4-2 所示，以后由该母版页生成的页面都将具有同样的结构和样式，只是 ContentPlace Holder 控件里面的内容会有所不同。

图 4-2　母版页的美化

4.1.4　任务评价

标　　准	掌 握 情 况
能理解母版页的工作原理	熟练□　基本□　模糊□
能创建母版页	熟练□　基本□　模糊□
能对母版页进行编辑	熟练□　基本□　模糊□
自 我 小 结	

4.1.5　任务拓展

读者可以参照本节中母版页的设计方法，自行完成 eBuy 商城前台网站母版页的设计，在设计时要尽量考虑用户操作的便利性以及结构的合理性。

任务 4.2　设计站点导航

4.2.1　任务分析

任务目标：站点导航是 Web 系统中一项非常实用的功能，在很多网站或者 Web 系统中都可以看到这项功能。在本任务中，将要在母版页上面添加相应的控件来实现站点导航

功能，同时还需要完成站点地图的设计。

完成标准：能在系统中实现站点导航功能。

应用手段：使用 SiteMapPath 控件和 TreeView 控件来实现。

4.2.2　相关知识

4.2.2.1　站点地图

若要使用 ASP.NET 站点导航，必须描述站点结构以便站点导航 API 和站点导航控件可以正确公开站点结构。默认情况下，站点导航系统使用一个包含站点层次结构的 XML 文件。不过，也可以将站点导航系统配置为使用其他数据源。

创建站点地图最简单的方法是创建一个名为 Web.sitemap 的 XML 文件，该文件按站点的分层形式组织页面。ASP.NET 的默认站点地图提供程序自动选取此站点地图。

尽管 Web.sitemap 文件可以引用其他站点地图提供程序、其他目录中的其他站点地图文件以及同一应用程序中的其他站点地图文件，但该文件必须位于应用程序的根目录中。

知识点小贴士

实现自定义的站点地图提供程序时，如果存储站点地图数据的文件的扩展名不是.sitemap，则会有潜在安全风险。默认情况下，ASP.NET 配置为阻止客户端下载具有已知文件扩展名（如 .sitemap）的文件。为帮助保护用户的数据，可将文件扩展名不是.sitemap 的所有自定义站点地图数据文件放入 App_Data 文件夹中。

下面的代码示例是演示站点地图如何查找一个三层结构的简单站点。url 属性可以以快捷方式“~/”开头，该快捷方式表示应用程序根目录。

```
<siteMap>
    <siteMapNode title="主页" description="Home" url="~/default.aspx">
        <siteMapNode title="产品" description="Our products" url="~/Products.aspx">
            <siteMapNode title="硬件" description="Hardware choices" url="~/Hardware
            .aspx" />
            <siteMapNode title="软件" description="Software choices" url="~/Software
            .aspx" />
        </siteMapNode>
    </siteMapNode>
</siteMap>
```

节点描述如下：

- siteMap：根节点，一个站点地图只能有一个根节点；
- siteMapNode：对应于页面的节点，一个节点对应一个页面；
- title：用于描述页面；
- url：用作链接文本的文本；
- description：用作文档和 SiteMapPath 控件中的工具提示，可以为空。

4.2.2.2 SiteMapPath 控件

SiteMapPath 控件会显示一个导航路径，此路径为用户显示当前页的位置，并显示返回到主页的路径链接。此控件提供了许多可供自定义链接的选项。

SiteMapPath 控件包含来自站点地图的导航数据，其中包括有关网站中的页的信息，如 URL、标题、说明和导航层次结构中的位置。若将导航数据存储在一个地方，则可以更方便地在网站的导航菜单中添加和删除项。

在 ASP 和 ASP.NET 的早期版本中，向网站添加一个页，然后在网站内的其他各页中添加指向该新页的链接时，必须手动添加链接，包括一个公共文件，或开发自定义导航功能。而 ASP.NET 2.0 版本包含一些导航控件，这些控件使得导航菜单的创建、自定义和维护变得很容易。

使用 SiteMapPath 控件无需代码和绑定数据就能创建站点导航。此控件可自动读取和呈现站点地图信息。但如果需要，也可以使用 SiteMapPath 控件来更改站点地图数据。

知识点小贴士

只有在站点地图中列出的页才能在 SiteMapPath 控件中显示导航数据。如果将 SiteMapPath 控件放置在站点地图中未列出的页上，该控件将不会向客户端显示任何信息。

4.2.2.3 TreeView 控件

TreeView 控件可按树形结构来显示分层数据，如目录或文件目录。TreeView 控件由一个或多个节点构成。树形结构中的每一项都称为“节点”。总共有 3 种不同的节点类型，分别为根节点、父节点和叶节点，其中根节点是没有父节点、但有一个或多个子节点的节点，父节点是具有一个父节点、且有一个或多个子节点的节点，叶节点是没有子节点的节点。

尽管一个典型的树形结构只有一个根节点，但 TreeView 控件允许向树形结构中添加多个根节点。如果希望在显示项列表的同时不显示单个主根节点（如在产品分类表中），此功能将十分有用。

每个节点都有一个 Text 属性和一个 Value 属性。Text 属性的值显示在 TreeView 控件中，而 Value 属性则用于存储有关该节点的任何附加数据，例如，传递给与节点关联的回发事件的数据。

TreeView 控件的常用属性如表 4-1 所示。

表 4-1 TreeView 控件的常用属性

属　　性	说　　明
CheckedNodes	声明被选择的单个或多个节点
ExpandDepth	声明 TreeView 控件展开的深度
Nodes	TreeNodeCollection 类型的节点集合
SelectedNode	当前被选择的节点
ShowCheckBoxes	声明是否显示复选框
ShowExpandCollapse	声明展开/折叠状态

续表

属　　性	说　　明
ShowLines	声明节点间是否以线连接
LevelStyles	指定每个层次的节点样式
NodeStyle	指定节点的默认样式
RootNodeStyle	指定根节点的样式
LeafNodeStyle	指定子节点的样式
SelectedNodeStyle	指定选定节点的样式
HoverNodeStyle	指定当鼠标移动到节点上方时的样式
ImageUrl Properties	指定表示展开/折叠的图片的 URL 路径

TreeView 控件的常用事件如表 4-2 所示。

表 4-2　TreeView 控件的常用事件

事　　件	说　　明
SelectedNodeChanged	当选择的节点发生改变时触发的事件
TreeNodeCollapsed	当分支被折叠时所触发的事件
TreeNodeExpanded	当分支被展开时所触发的事件
TreeNodeDataBound	当节点被绑定到数据源时所触发的事件

TreeView 控件可以直接从 XML 文件中获取节点的信息来进行显示，XML 文件的格式和站点地图的格式基本一样。

4.2.3　任务实施

1．创建站点地图

（1）右击“解决方案资源管理器”中的网站，在弹出的快捷菜单中选择“添加新项”|“站点地图”命令，打开如图 4-3 所示的“添加新项”对话框。

图 4-3　“添加新项”对话框

（2）选择“站点地图”选项，单击“添加”按钮，完成站点地图的创建。此时便可以在“解决方案资源管理器”中看到 web.sitemap 文件，如图 4-4 所示。

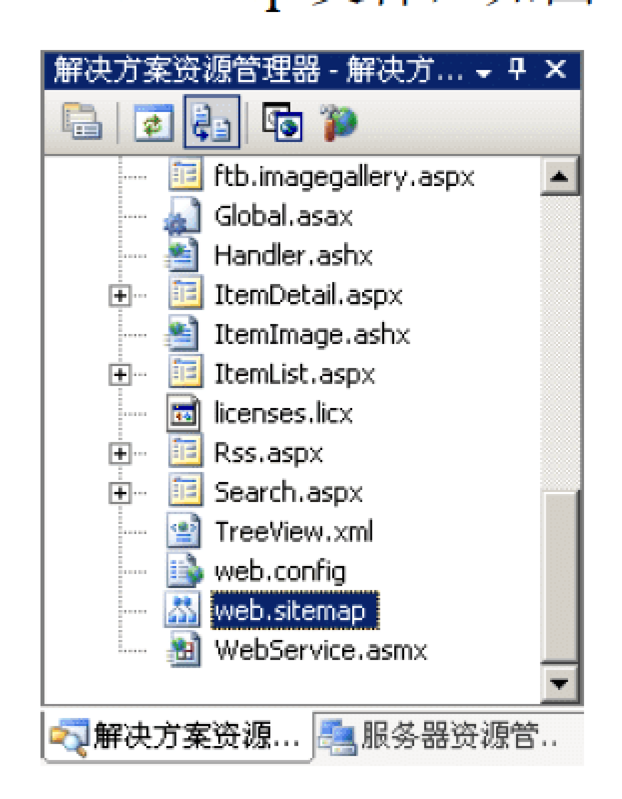

图 4-4　站点地图文件

（3）双击“解决方案资源管理器”中的 web.sitemap 文件，就可以对站点地图文件进行编辑，其主要代码如下：

```
<?xml version="1.0" encoding="utf-8" ?>
    <siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
        <siteMapNode Id="" url="~\Admin\default.aspx" title="管理员后台" description="">
            <siteMapNode url="" title="用户管理" description="">
                <siteMapNode url="~\Admin\ListAllUsers.aspx" title="管理用户" description="" />
                <siteMapNode url="~\Admin\UserStatue.aspx" title="状态管理" description="" />
                <siteMapNode url="~\Admin\UserDetails.aspx" title="修改用户资料" description="" />
            </siteMapNode>
        </siteMapNode>
</siteMap>
```

从上述代码可以看出，节点“管理员后台”下面包含有一个子节点“用户管理”，而“用户管理”节点下面则又包含有 3 个子节点，分别是“管理用户”、“状态管理”和“修改用户资料”。在进行页面访问时，站点导航控件将按照这个 XML 中的层次结构来显示站点导航。例如，当页面“Admin\UserDetails.aspx”被访问时，站点导航控件的内容就显示为“管理员后台>用户管理>修改用户资料”。

2．创建用于 TreeView 显示的 XML 文件

（1）右击网站下的“Admin 文件夹”，在弹出的快捷菜单中选择“添加新项”命令，打开“添加新项”对话框。选择“XML 文件”选项，并将其命名为“admin_menu.xml”，如图 4-5 所示。

（2）单击“添加”按钮完成 XML 文件的添加。这个 XML 文件将会被用于 TreeView 控件的内容显示，在此将 TreeView 控件用作页面左边的树形菜单。编辑“admin_menu.xml”文件，其主要内容如下：

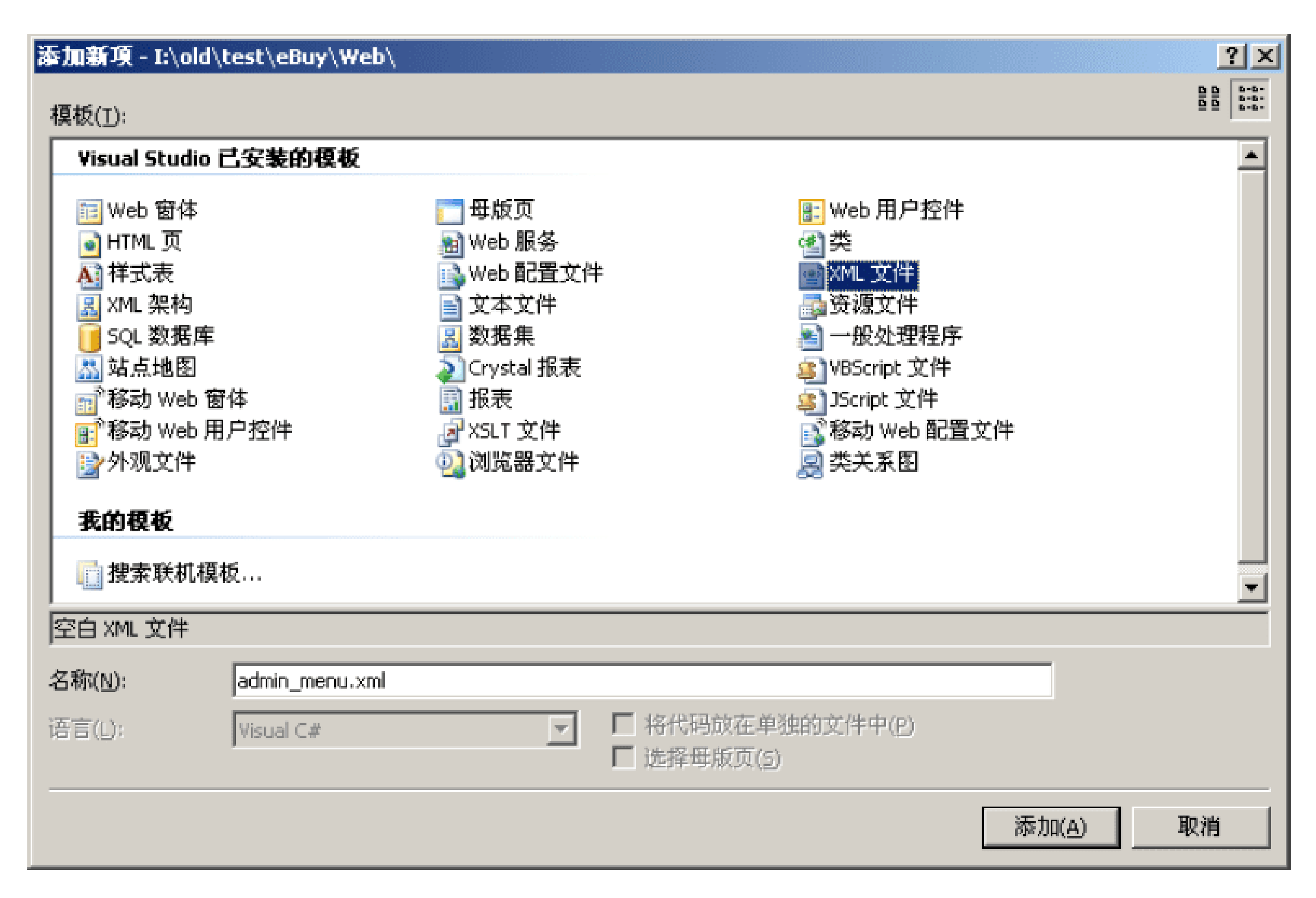

图 4-5　选择“XML 文件”

```
<?xml version="1.0" encoding="utf-8"?>
    <siteRoot Id="root" url="" title="管理员控制面板" description="">
        <siteMapNode url="" title="用户管理" description="">
            <siteMapNode url="~\Admin\ListAllUsers.aspx" title="用户管理" description="" />
            <siteMapNode url="~\Admin\UserStatue.aspx" title="状态管理" description="" />
        </siteMapNode>
        <siteMapNode url="" title="商品及分类管理" description="">
            <siteMapNode url="~\Admin\AddItemsCategory.aspx" title="添加商品分类" description="" />
            <siteMapNode url="~\Admin\ListItemsByCategory.aspx" title="为商品分类" description="" />
            <siteMapNode url="~\Admin\ListOfItems.aspx" title="商品列表" description="" />
        </siteMapNode>
        <siteMapNode url="" title="订单管理" description="">
            <siteMapNode url="~\Admin\CheckOrders.aspx" title="审核订单" description="" />
        </siteMapNode>
        <siteMapNode url="~\Admin\LoginOut.aspx" title="退出" description="管理员退出">
        </siteMapNode>
    </siteRoot>
```

这样，树形菜单中的内容以及内容的层次都将按照这个 XML 中的内容和层次进行相应的显示。

3．添加 SiteMapPath 控件和 TreeView 控件

由于 SiteMapPath 控件和 TreeView 控件需要显示在所有的页面上，因此可以直接将它们添加到母版页上面。在母版页的相应位置分别添加 SiteMapPath 控件和 TreeView 控件，并绑定它们的数据源，如图 4-6 所示。在指定好数据源后，使用 TreeView 控件的树形菜单可以直接将 admin_menu.xml 中的内容获取出来并进行显示。

图 4-6　添加控件

4.2.4　任务评价

标　　准	掌 握 情 况
能创建和编辑站点地图	熟练□　基本□　模糊□
能使用 SiteMapPath 控件实现站点导航	熟练□　基本□　模糊□
能使用 TreeView 控件实现树形菜单	熟练□　基本□　模糊□
自 我 小 结	

4.2.5　任务拓展

参照本节中的内容，读者可自行完成如下两项任务：使用 SiteMapPath 控件实现前台页面的站点导航，以及使用 TreeView 控件实现前台产品分类的树形菜单。

本章小结

本章通过后台管理系统的母版页和站点导航的设计，阐述了使用母版页来统一界面设计样式和结构的一般方法，包括母版页的创建以及母版页的设计等内容。另外，本章也阐述了使用 SiteMapPath 控件以及站点地图文件来实现站点导航的方法，同时还讲解了通过配合使用自定义的 XML 文件和 TreeView 控件来生成树形菜单的方法。

自我检测

一、选择题

1．母版页的扩展名为（　）。

A．.asax　　B．.aspx

C．.master　　D．.ascx

2．有关导航控件的说法正确的是（　）。

A．作为服务器控件，导航控件将生成浏览器可执行的 JavaScrip 脚本和 HTML 代码。

B．SiteMapPath 可以使用 XML 文件格式作为数据源。

C．TreeView 控件所使用的数据源一定是 XML 文件格式的。

D．TreeView 控件的 TextField 属性用于链接对应的字段或元素值。

3．实现控件在页面上任意位置的放置可以通过的操作为（　）。

A．鼠标拖动　　B．设置布局方式为相对

C．设置布局方式为绝对　　D．使用表格

4．关于母版页的说法，正确的是（　）。

A．一个站点只能有一个母版页

B．一个内容页对应母版中的一个位置

C．内容页相当于 HTML 中的 iframe 页，浏览地址显示母版页地址

D．母版页的后缀名是.master

5．关于站点地图文件的说法，正确的是（　）。

A．站点地图文件是一个 HTML 文件

B．站点地图文件中可以有多个 siteMap 标签

C．站点地图文件中可以有多个 siteMapNode 标签

D．站点地图文件反映的就是项目中页面文件的相对关系

6．The following statements about site map file, which is right（　）.

A．Site map file is HTML file

B．Site map file can contain several <siteMap> label

C．Site map file reflects the relationship between page files in one project

D．Site map file can contain several <siteMapNode> label

二、简答题

1．在 Master 和 Content 页面之间，哪些值必须相互匹配？

2．使用母版页来进行页面设计有哪些好处？

3．简述 TreeView 控件使用数据源应注意的地方。

第 5 章　显 示 数 据

技能目标

1．能使用 GridView 控件来完成数据的展示。
2．能使用 ObjectDataSource 控件绑定业务逻辑层返回的对象。
3．能使用 DetailsView 控件完成数据的详细展示。

相关词汇

英 文 单 词	中 文 含 义	英 文 单 词	中 文 含 义
details		bind	
list		grid	
data source		object	
module			

工作任务

任务 5.1　显示列表信息
任务 5.2　显示详细信息

任务 5.1　显示列表信息

5.1.1　任务分析

任务目标：在管理系统中经常需要将数据库中的数据以一定的方式对用户进行显示，以方便用户查看或者管理数据。ASP.NET 中提供了多种不同的数据显示方式，在本任务中，将以用户管理模块为例，使用 ASP.NET 提供的数据显示控件将用户管理模块所需要的数据进行显示，其效果如图 5-1 所示。

完成标准：能够按格式显示用户信息列表。

应用手段：使用 GridView 控件来实现。

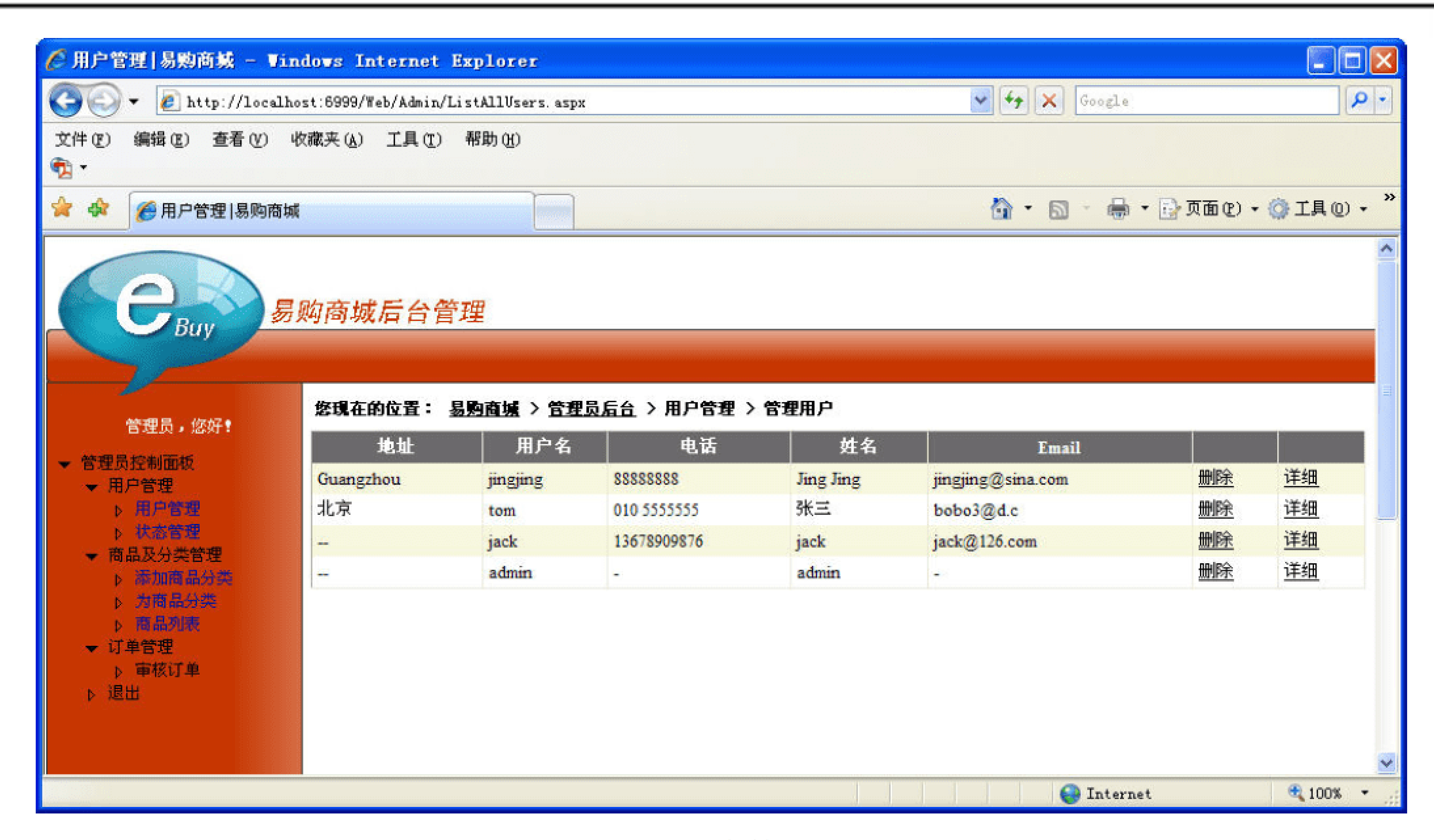

图 5-1　用户管理效果

5.1.2　相关知识

5.1.2.1　数据绑定

使用 ASP.NET 数据绑定，可以将任何服务器控件绑定到简单的属性、集合、表达式或有数据返回的方法中。使用数据绑定，在数据库中使用数据或通过其他方法使用数据时，会具有更大的灵活性。

ASP.NET 有两种数据绑定方式。

1．编码指定数据源

编码指定数据源方式就是采用编写代码在程序运行中动态绑定数据源。例如：

```
this.gvMain.DataSource=UserManager.GetAllUsers();
this.gvMain.DataBind();
```

2．使用数据源控件

ASP.NET 包含一些数据源控件，这些数据源控件允许使用不同类型的数据源，如数据库、XML 文件或中间层业务对象。数据源控件连接到数据源，从中检索数据，并可以无需代码使得其他控件绑定到数据源。数据源控件还支持修改数据。

5.1.2.2　数据源控件

.NET Framework 包含支持不同数据绑定方案的数据源控件。表 5-1 给出了内置的数据源控件及相关说明。

表 5-1　数据源控件

数据源控件	说　明
ObjectDataSource	允许使用业务对象或其他类，以及创建依赖中间层对象管理数据的 Web 应用程序。支持对其他数据源控件不可用的高级排序和分页方案
SqlDataSource	允许使用 Microsoft SQL Server、OLE DB、ODBC 或 Oracle 数据库。与 SQL Server 一起使用时支持高级缓存功能。当数据作为 DataSet 对象返回时，此控件还支持排序、筛选和分页
AccessDataSource	允许使用 Microsoft Access 数据库。当数据作为 DataSet 对象返回时，支持排序、筛选和分页
XmlDataSource	允许使用 XML 文件，特别适用于分层的 ASP.NET 服务器控件，如 TreeView 或 Menu 控件。支持使用 XPath 表达式来实现筛选功能，并允许对数据应用 XSLT 转换。XmlDataSource 允许通过保存更改后的整个 XML 文档来更新数据
SiteMapDataSource	结合 ASP.NET 站点导航使用

5.1.2.3　GridView 控件

显示表格数据是软件开发中要反复执行的一项任务。ASP.NET 提供了许多工具，用于在网格中显示表格数据，如 GridView 控件。使用 GridView 控件，可以显示、编辑和删除来自多种不同的数据源（包括数据库、XML 文件和公开数据的业务对象）的数据。表 5-2 中列出了 GridView 控件的常用属性。

表 5-2　GridView 控件的常用属性

属　性	说　明
AllowPaging	指示该控件是否支持分页
AllowSorting	指示该控件是否支持排序
AutoGenerateColumns	指示是否自动为数据源中的每个字段创建列。默认为 true
DataMember	指示一个多成员数据源中的特定表绑定到该网格。该属性与 DataSource 结合使用。如果 DataSource 有一个 DataSet 对象，则该属性包含要绑定的特定表的名称
DataSource	获得或设置包含用来填充该控件的值的数据源对象
DataSourceID	指示所绑定的数据源控件
PageSize	获取或设置 GridView 控件在每页上所显示记录的数目
PageCount	获取在 GridView 控件显示数据源记录所需的页数
PageIndex	获取或设置当前显示页的索引
SortDirection	获得列的当前排序方向
SortExpression	获得当前排序表达式

5.1.3　任务实施

1．添加数据显示页面

由于后台管理页面都需要从母版页生成，因此，在 Admin 文件夹下面添加 ListAllUsers.aspx 页面时一定要选中“添加新项”对话框右下角的“选择母版页”复选框，如图 5-2 所

示，然后选择母版页 admin.master，完成内容页的生成。

图 5-2　“添加新项”对话框

2．添加 GridView 控件

（1）在 Visual Studio 2005 的工具栏中将 GridView 控件拖入到页面的相应位置，这时需要选择“新建数据源”选项，如图 5-3 所示。

图 5-3　添加 GridView 控件

（2）打开“数据源配置向导”对话框，选择“对象”选项，创建对象数据源（ObjectDataSource），并为数据源指定 ID 为“odsUsers”，如图 5-4 所示。

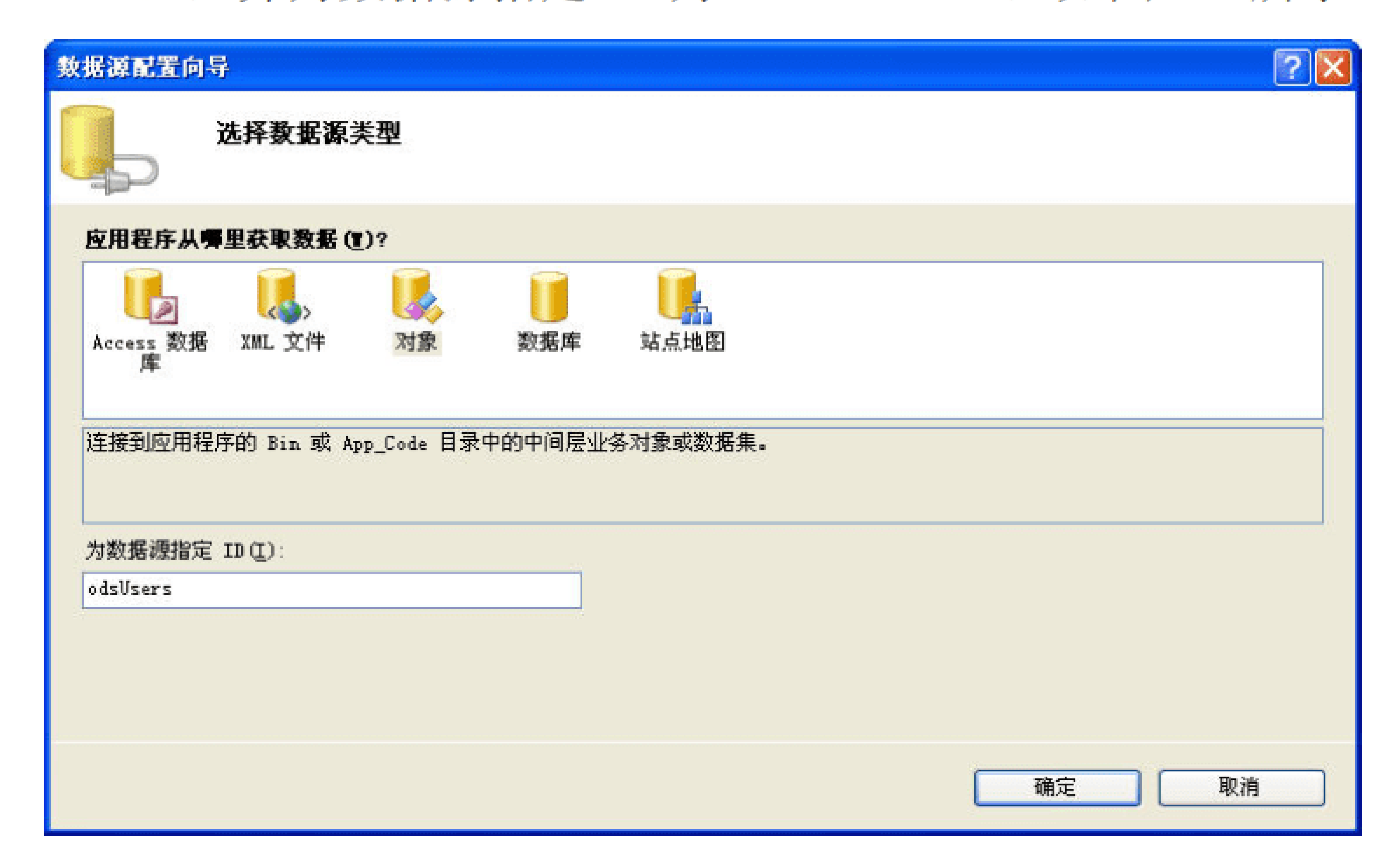

图 5-4　选择数据源类型

（3）单击“确定”按钮，弹出“配置数据源”对话框，选择业务逻辑层中用来操作 User 实体的类 BLL.UserManager，如图 5-5 所示。

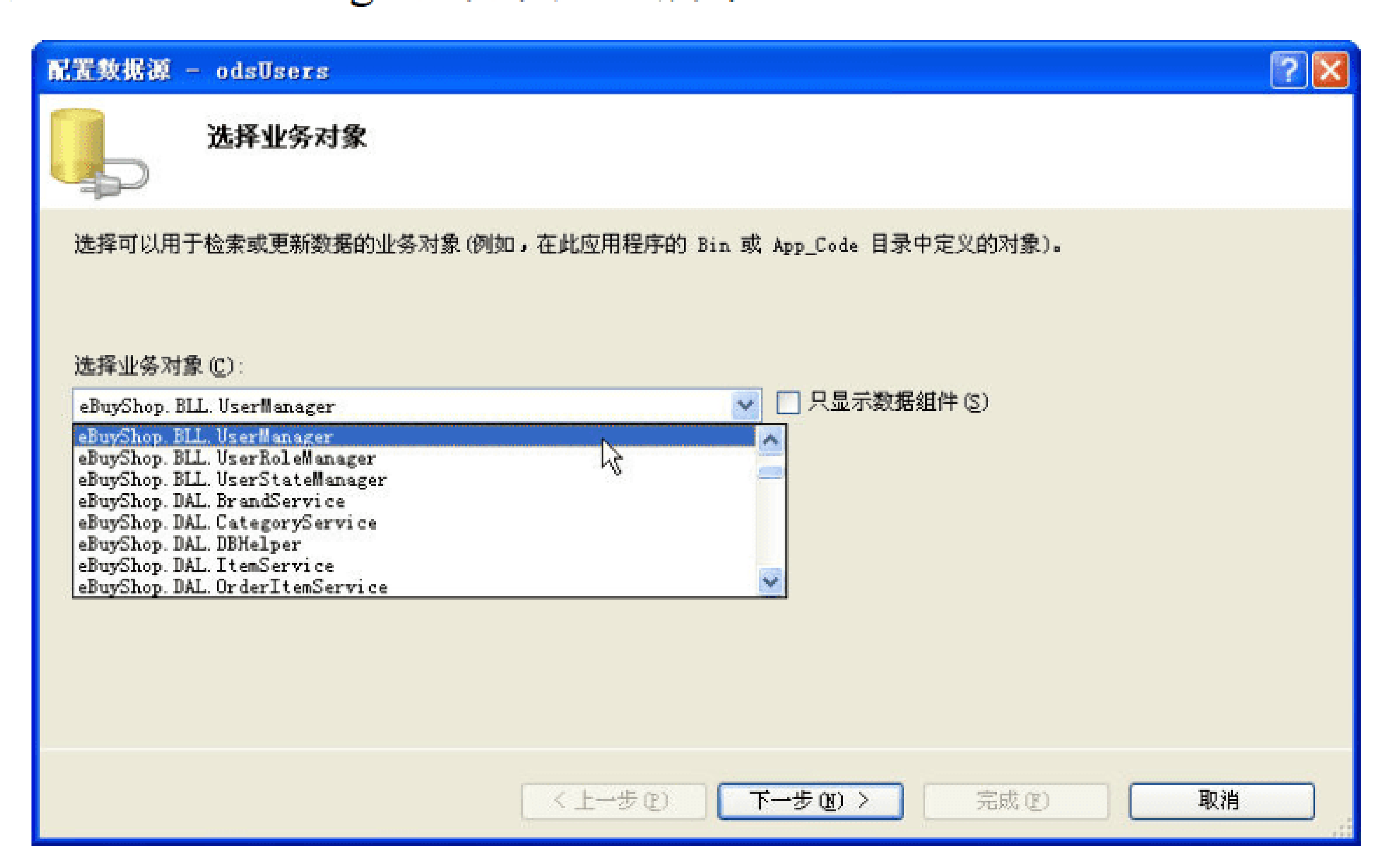

图 5-5　选择业务对象

（4）单击“下一步”按钮，进入“定义数据方法”界面，在此设置需要的增、删、改以及查方法。因为需要显示和删除相关用户信息，所以在 SELECT 选项卡中选择 GetAllUsers()方法，如图 5-6 所示。

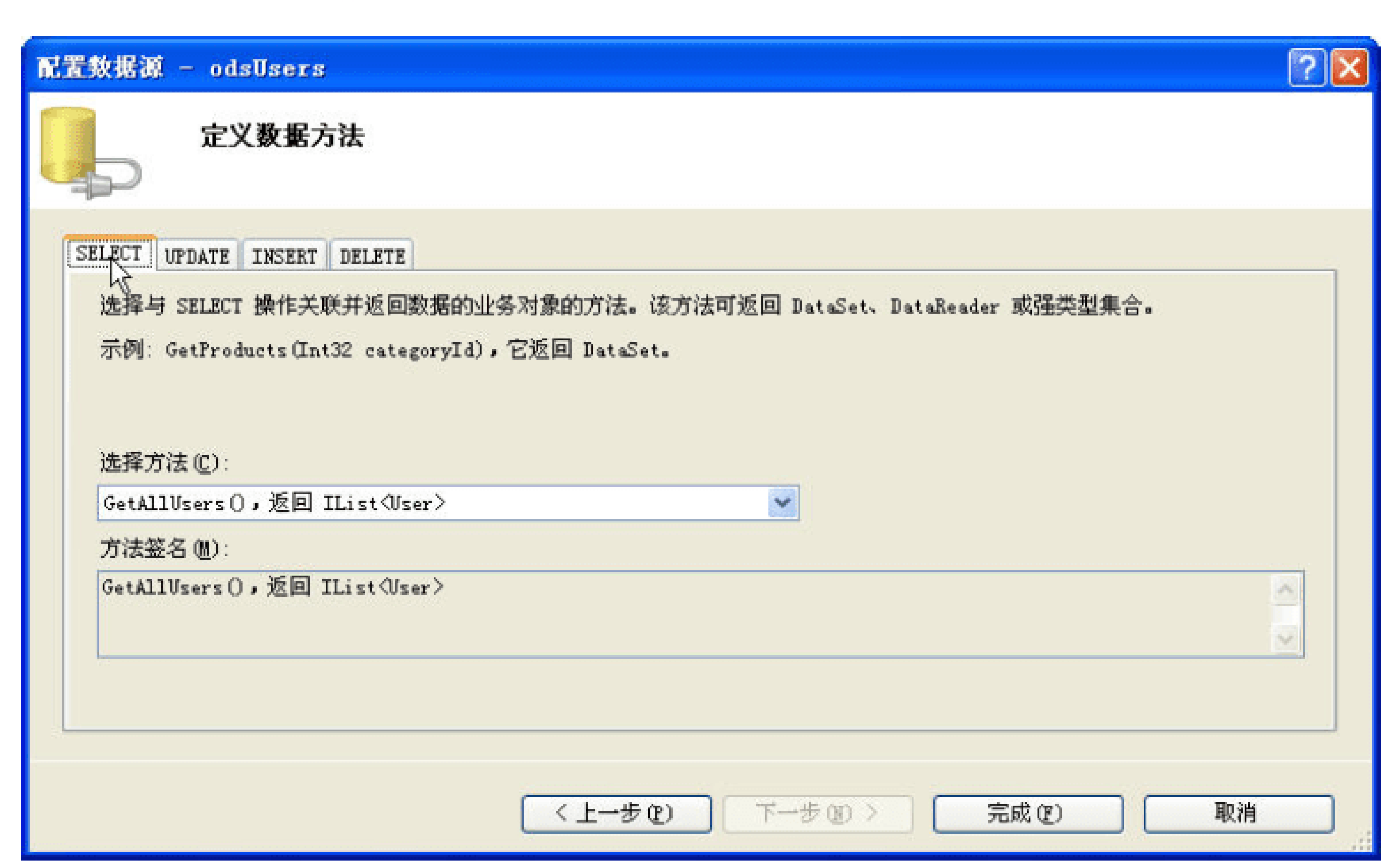

图 5-6　定义数据方法

（5）选择 DELETE 选项卡，选择 DeleteUser(User user)方法，如图 5-7 所示。

对话框中所提供的方法实际上都来源于业务逻辑层中的 BLL.UserManager 类，通过这种方式，对象数据源就可以直接调用封装在业务逻辑层中的方法来实现相应的操作。

（6）单击“完成”按钮，就可以实现对所有用户信息的显示，如图 5-8 所示。

（7）编辑 GridView 控件。通过对 GridView 控件列的修改及数据绑定等工作，最终效果如图 5-9 所示。

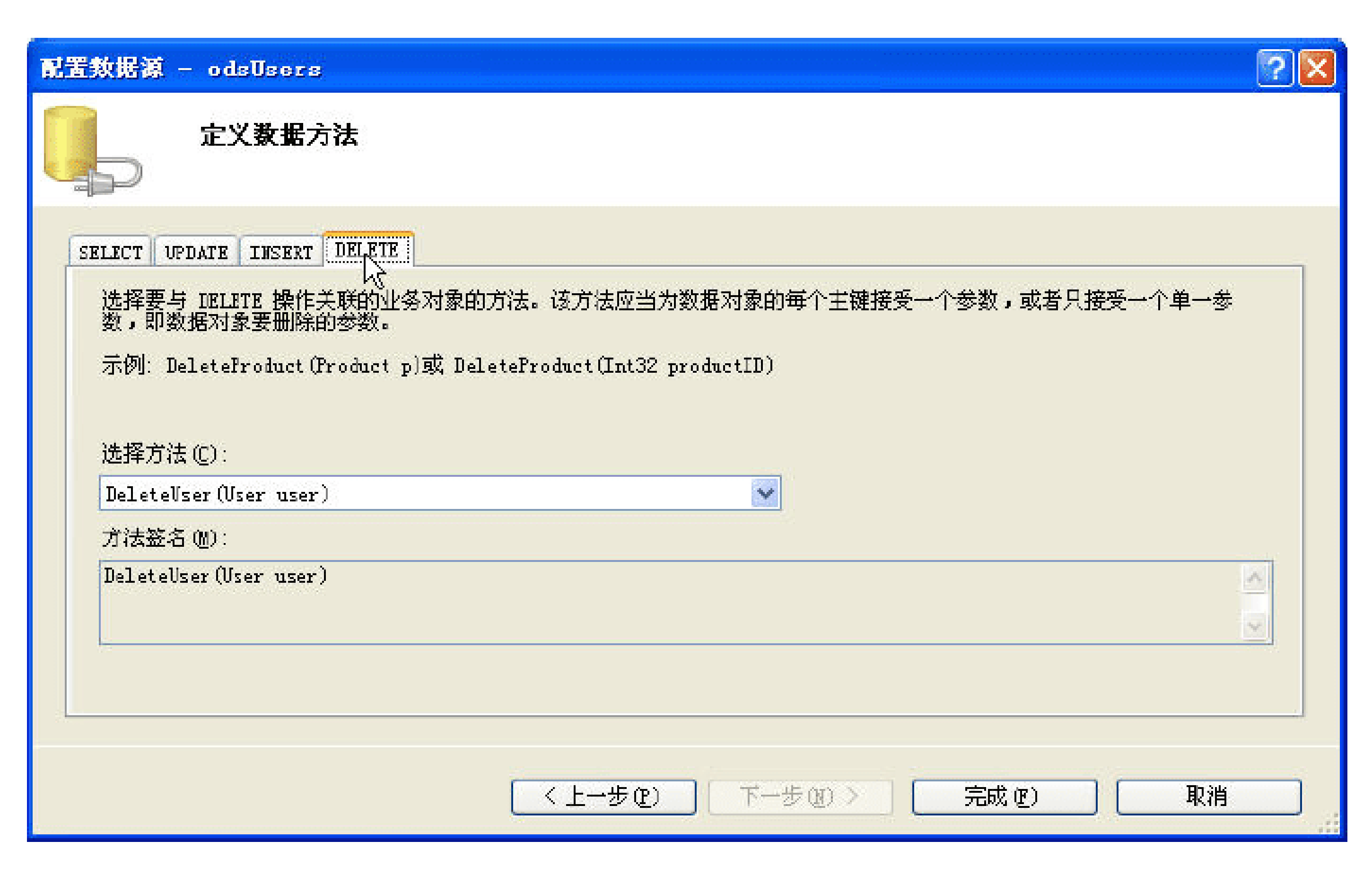

图 5-7　定义数据方法

图 5-8　显示用户信息

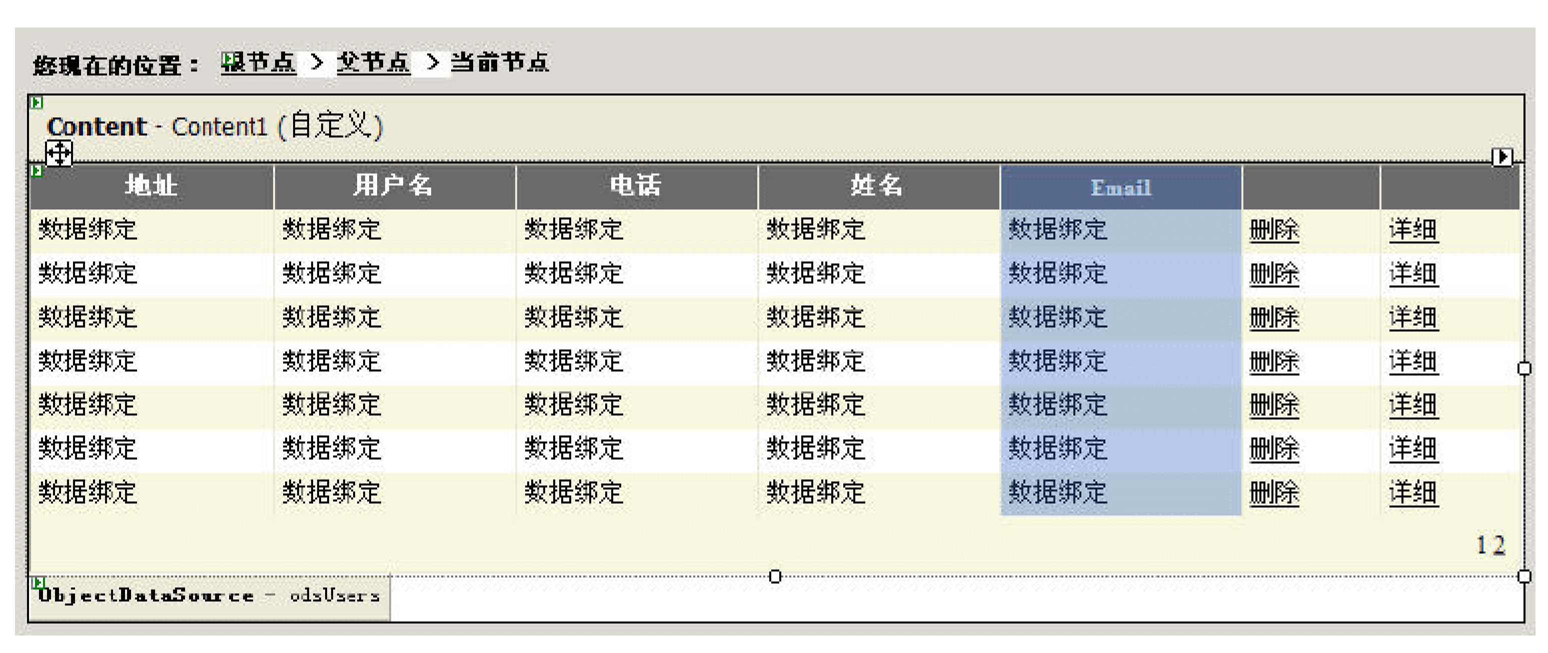

图 5-9　GridView 控件使用效果

5.1.4　任务评价

标　　准	掌 握 情 况
能对对象数据源进行设置	熟练□　基本□　模糊□
能对 GridView 控件进行模板编辑	熟练□　基本□　模糊□
能使用 GridView 控件进行数据显示	熟练□　基本□　模糊□
自 我 小 结	

5.1.5 任务拓展

本节中使用 GridView 控件实现了用户信息的显示和删除，读者可参照上述内容在其基础上通过对 GridView 控件及对象数据源的修改来实现用户信息的修改功能。

任务 5.2 显示详细信息

5.2.1 任务分析

任务目标：列表通常用来显示批量的数据，如果要查看某一条数据的详细信息则需要打开该条数据行进行显示。在本任务中，将采用数据显示控件来显示单条数据详细信息，用户可以通过单击数据列表中的某一条数据来查看其详细信息，如图 5-10 所示。

图 5-10 具体用户信息

完成标准：能够按格式显示选定用户的详细信息。

应用手段：使用 DetailsView 控件来实现。

5.2.2 相关知识

使用 DetailsView 控件，可以逐一显示、编辑、插入或删除其关联数据源中的记录。默认情况下，DetailsView 控件将逐行单列显示记录的各个字段。DetailsView 控件通常用于更新数据或插入新记录，并且通常在主/详细方案中使用，在这些方案中，主控件选中的记录决定了在 DetailsView 控件中显示的记录。例如，采用 GridView 控件来作为主控件显示所

有用户的数据列表，采用 DetailsView 控件作为详细控件来显示主控件中选中的某一个用户的详细信息。即使 DetailsView 控件的数据源公开了多条记录，该控件每次也只会显示一条数据记录。表 5-3 中列出了 DetailsView 控件的常用属性和事件。

表 5-3　DetailsView 控件的常用属性和事件

属性或事件	说　　明
DefaultMode 属性	设置或获取控件的状态（模式）。该属性为枚举值，分为 ReadOnly（显示）、Edit（修改）和 Insert（添加）
DataKey 属性	数据的主键
DataKeyNames 属性	设置或获取一个字符串，该字符串包含数据源中间的组合
ItemInserting 事件	单击“添加”按钮，执行添加方法之前执行
ItemInserted 事件	单击“添加”按钮，执行添加方法之后执行
ItemUpdating 事件	单击“更新”按钮，执行更新方法之前执行
ItemUpdated 事件	单击“更新”按钮，执行更新方法之后执行

5.2.3　任务实施

1．添加页面

与添加 ListAllUsers.aspx 页面类似，新页面 UserDetails.aspx 也是在 Admin 文件夹下面添加，该页面也需要从母版页中生成。

2．添加 DetailsView 控件

（1）DetailsView 控件和 GridView 控件的使用方法比较相似。从工具栏中选中 DetailsView 控件，并将它拖放到页面上，同样设置其数据源为 ObjectDataSource。与 GridView 不同的是，这里要显示某一个用户的信息，因此查询方法就需要选择 GetUserById 选项，如图 5-11 所示。

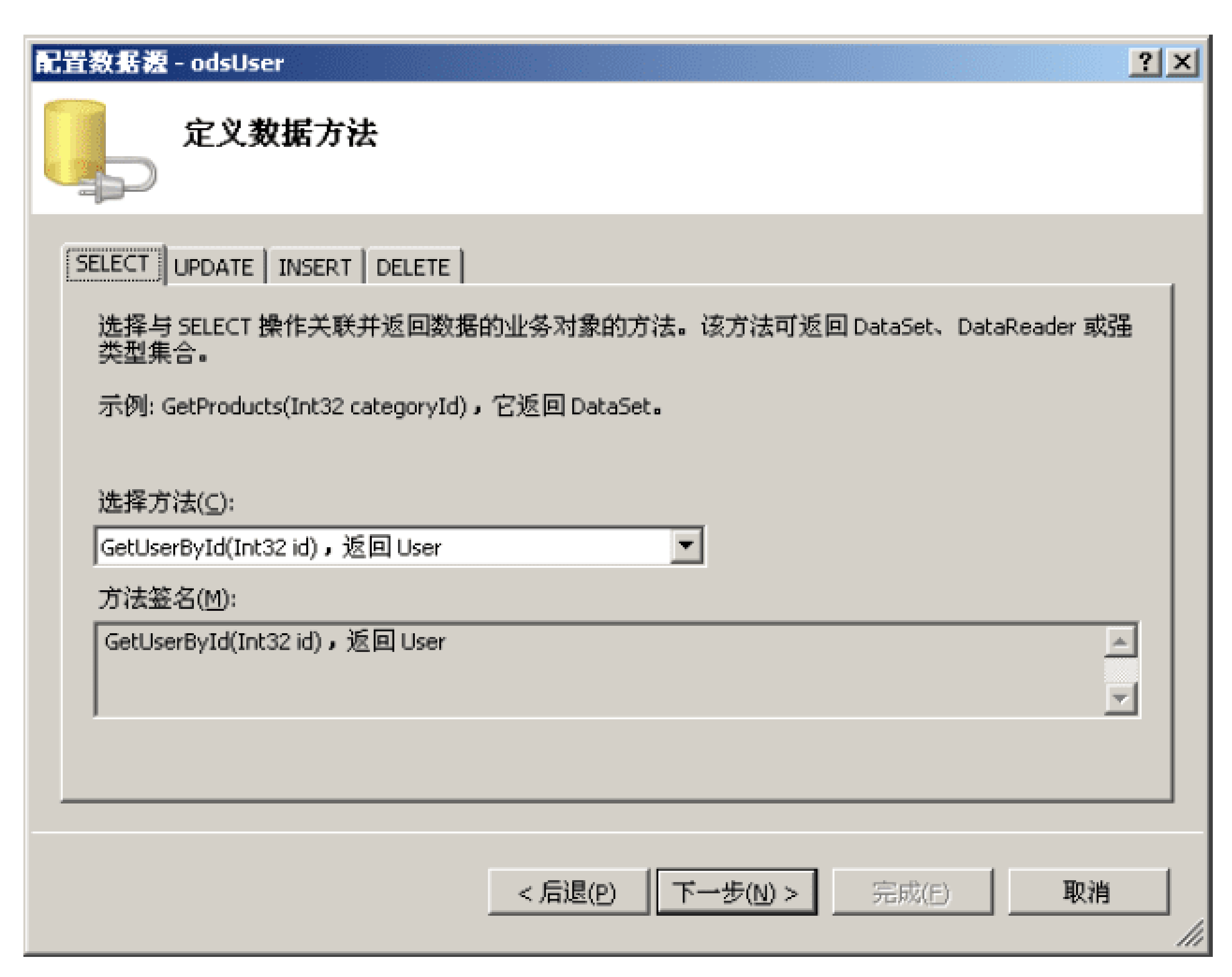

图 5-11　“定义数据方法”对话框

（2）单击“下一步”按钮，进入如图 5-12 所示的“定义参数”对话框。由于之前选择的方法是带有参数的，因此在这一步中需要指明参数的来源和初始值。这里选择参数源为 QueryString，表示参数是从上一页面地址栏上传递过来的。参数的名称为“id”，初始值为 0。

图 5-12 “定义参数”对话框

知识点小贴士

定义参数时有如下的集中参数源可供选择：

Cookie：表示参数是从保存在客户机上的 Cookie 中获取。

Control：表示参数是直接通过控件进行传递。

Form：表示参数是通过页面上的表单进行传递。

Profile：表示从配置文件中获取参数。

QueryString：表示从请求查询字符串中获取参数。

Session：表示从 Session 对象中获取参数。

3．编辑 DetailsView 控件

通过编辑模板对 DetailsView 控件进行显示样式编辑，最后的效果如图 5-13 所示。

图 5-13 DetailsView 控件使用效果

5.2.4　任务评价

标　　准	掌 握 情 况
能设置带参数的对象数据源	熟练□　基本□　模糊□
能对 DetailsView 控件进行模板编辑	熟练□　基本□　模糊□
能使用 DetailsView 控件进行数据显示	熟练□　基本□　模糊□
自 我 小 结	

5.2.5　任务拓展

读者可以采用类似的方法来实现对商品信息的显示，包括有商品信息的列表显示、商品信息的详细信息显示。

本章小结

本章通过实现用户信息的列表和详细显示，阐述了通过使用数据显示控件来实现数据显示的基本方法。主要有使用 GridView 控件来实现数据的列表显示、使用 DetailsView 控件来实现单个对象的详细信息显示，同时也介绍了数据源的配置方式以及带参数数据源的配置方式。

自我检测

一、选择题

1．GridView 设置分页后，默认显示的记录条数为（　）。

A．5　　B．10

C．15　　D．25

2．GridView 控件的基类是（　）。

A．ListControl　　B．CompositeDataBoundControl

C．BaseGridriew　　D．HierarchicalDataBoundControl

3．下列控件中不支持插入记录的是（　）。

A．GridView 控件　　B．DetailsView 控件

C．DropDownList 控件　　D．都不支持

4．下列控件只提供可编辑空白区域的是（　）。

A．GridView 控件　　B．DetailsView 控件

C．DropDownList 控件　　D．都可以

5．如果要在 GridView 控件中显示商品的图片，说法正确的是（　）。

A．可以使用模板　　B．可以使用 ImageField

C．只能通过编码实现　　D．无法显示

6．Which is the base class of GridView control (　).

A．ListControl　　B．BaseGridriew

C．HierarchicalDataBoundControl　　D．CompositeDataBoundControl

二、简答题

1．简述主要有哪些数据源控件？它们分别有什么优缺点？

2．GridView 控件在方便性和灵活性方面提供了哪些处理方式？

3．简述 DetailsView 控件的使用流程。

第 6 章　添加和更新数据

技能目标

1．能使用 ObjectDataSource 等控件实现数据更新。
2．能使用验证控件对客户端输入的数据进行验证。
3．能编写代码实现数据的添加和更新。
4．会使用在数据添加和更新中常用的辅助控件。

相关词汇

英 文 单 词	中 文 含 义	英 文 单 词	中 文 含 义
expression		summary	
verification		range	
compare		validation	
regular		custom	

工作任务

任务 6.1　实现注册验证
任务 6.2　修改商品信息

任务 6.1　实现注册验证

6.1.1　任务分析

任务目标：使用验证控件实现用户注册的数据验证。用户在注册时需要从客户端输入一定量的数据，如用户名、密码等。根据具体要求对这些数据进行相应的验证，例如，用户名不能为空，或者还有长度的要求等。通过注册验证，可以使用户提交到服务器端的数据尽可能地符合要求。

完成标准：能够对用户注册中的各个数据项进行验证，完成效果如图 6-1 所示。

应用手段：使用 ASP.NET 中提供的多个数据验证控件来完成不同数据项的验证。

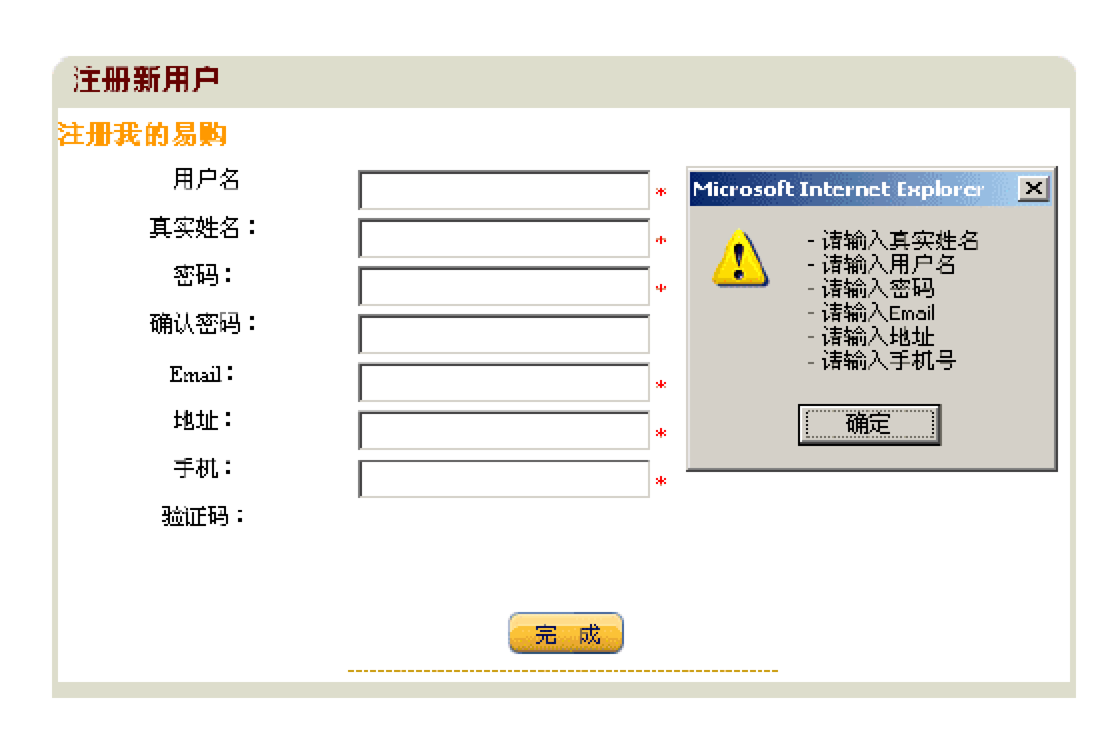

图 6-1　用户注册的验证

6.1.2　相关知识

数据验证是一项非常重要并且很常用的技术。Web 应用系统中，很多功能都基于用户客户端与服务器端的交互，与此同时也会有各种各样的数据在客户端与服务器端之间进行交换。数据验证的目的就在于通过一定的技术手段，尽可能地减少来自客户端的不合法的数据，以减少一些可能发生的错误。

数据验证有很多方式。在 3.5 节中已经制作过一个用户登录页面，为避免用户提交空白的用户名和密码，这里编写了一段代码对用户名和密码这两个数据项进行判断，看它们是否为空。这其实就是一种数据验证的方式，但这种方式有它的局限性，因为该判断代码是在服务器端运行的，也就是说用户单击“提交”按钮后，数据需要先提交到服务器端，然后才能进行判断，处理后的结果还需要再从服务器端返回到客户端。

另外一种验证方式是在客户端进行的，可以通过编写 JavaScript 脚本在客户端编写验证代码，这样，用户在进行表单提交时，会激发客户端验证脚本，使其直接在客户端对数据项进行判断。但是，JavaScript 验证方式也有些缺点：

（1）JavaScript 脚本不够安全，某些验证代码可以人为地绕开。

（2）JavaScript 脚本编写代码比较复杂，而且没有很好的调试方式。

ASP.NET 提供了一系列的验证控件，来完成各种不同类型数据项的验证，包括 5 种验证控件和 1 个汇总控件，这几种验证控件各自实现了不同的验证功能，它们之间的关系如图 6-2 所示。

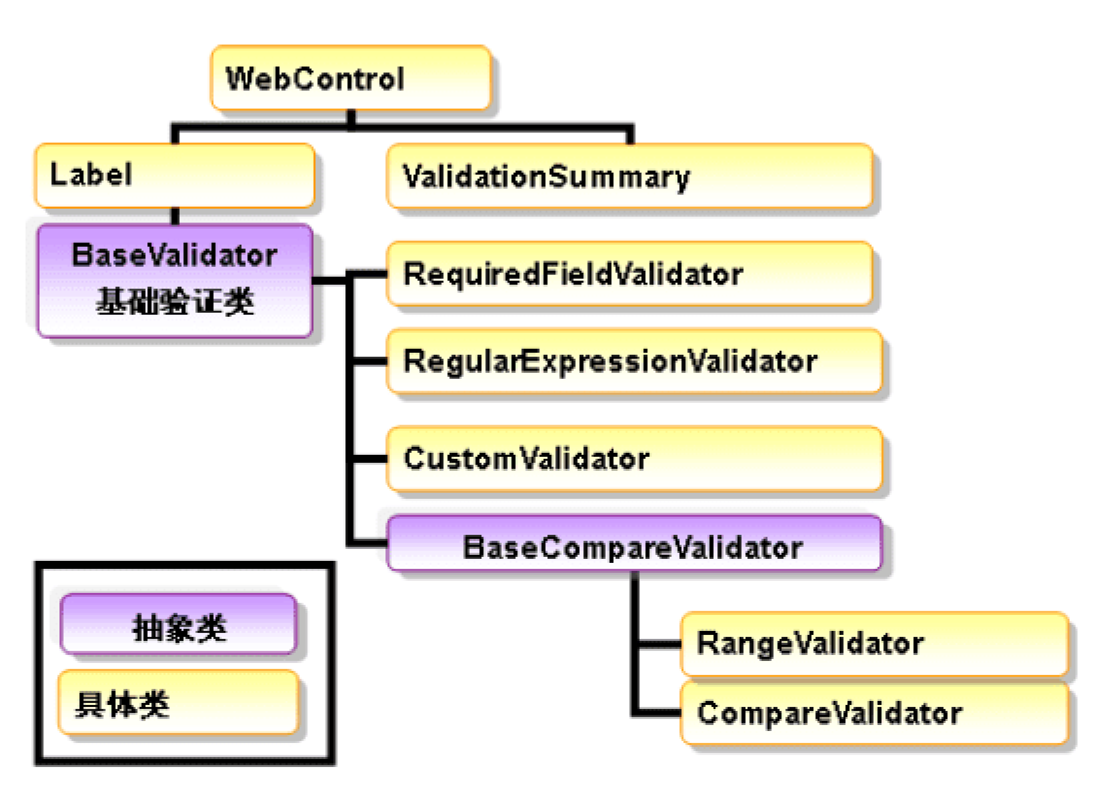

图 6-2　验证控件之间的关系

在 IDE 工具箱中的“验证”展开项中可以找到这些验证控件，如图 6-3 所示。

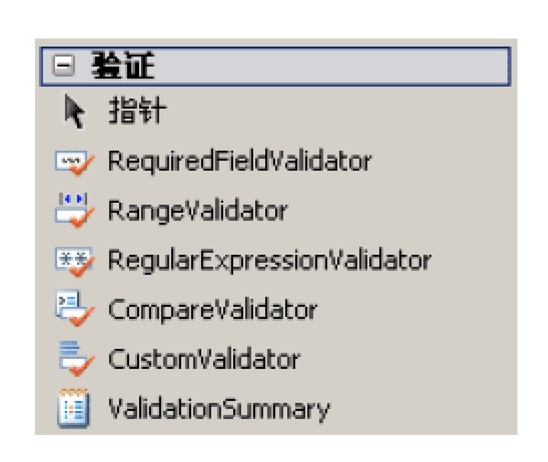

图 6-3　验证控件

下面逐个介绍图 6-3 中的几种验证控件的用法。

1．RequiredFieldValidator 控件

从 RequiredFieldValidator 的字面意思可以看出，它是用来验证用户提供有效的输入的控件，即进行非空的验证。RequiredFieldValidator 是最基本也最常用的一个验证控件，通常和其他验证控件一起实现有效性验证。表 6-1 中列出了 RequiredFieldValidator 控件的常用属性。

表 6-1　RequiredFieldValidator 控件的常用属性

属　　性	描　　述
ControlToValidate	要进行验证的控件 ID
Text	当验证失败时显示的信息
ErrorMessage	当验证失败时，在 ValidationSummary 验证汇总控件中显示的文本

2．CompareValidator 控件

CompareValidator 控件功能通过比较用户输入的数据验证输入是否符合要求。此控件最常见的用途是用来验证用户在注册时两次输入的密码是否相同。当然 CompareValidator 控件不仅可以判断一个控件中的值和另外一个控件的值是否相等，还可以比较它们的大小。表 6-2 列出了 CompareValidator 控件的常用属性。

表 6-2　CompareValidator 控件的常用属性

属　　性	说　　明
ControlToValidate	要进行验证的控件 ID
ControlToCompare	用来与要验证的控件进行比较的控件 ID
Type	设置比较的数据类型，如字符串、整型等
Operator	设置比较运算符，如等于、大于等于、大于等。默认为等于
ValueToCompare	用来直接进行比较的值

知识点小贴士

基于 CompareValidator 的属性，它也可以用来验证数据类型，例如，用户生日是一个日期，可以使用 CompareValidator 控件来进行验证。当然，在这种情况下，需要移除 ValueToCompare 和 ControlToCompare 的值，并设置 Type 类型为 Date，同时设置 Operator 值为 DataTypeCheck。

3. RangeValidator 控件

RangeValidator 是用来进行范围验证的一个控件。例如，人的年龄一般都设置在 0～100 岁。表 6-3 列出了 RangeValidator 控件的常用属性。

表 6-3 RangeValidator 控件的常用属性

属　　性	描　　述
ControlToValidate	要进行验证的控件 ID
MaximumValue	范围的上界
MinimumValue	范围的下界
Type	验证的类型（字符串、整型、双精度、日期和货币 5 种类型）

4. RegularExpressionValidator 控件

RegularExpression 就是通常所说的正则表达式，RegularExpressionValidator 是采用正则表达式的方式来进行数据验证的控件。例如，在进行电子邮件地址的验证时，就要求用户输入的数据中包含有“@”符号和“.”号，并且对它们之间以及和其他字母之间的排列顺序也有要求，这时采用正则表达式的方式就可以很方便地进行验证。RegularExpressionValidator 中内置了几种常用的正则表达式，只需选择控件的 RegularExpression 属性就可以打开正则表达式编辑器，然后选择要使用的正则表达式即可。

知识点小贴士

常用的正则表达式有：

非负整数：^\d+$。

正整数：^[0-9]*[1-9][0-9]*$。

中文字符：[\u4e00-\u9fa5]。

货币（小数点后两位）：\d+(\.\d\d)。

关于正则表达式的基本语法，读者可以查阅相关的资料。

5. CustomValidator 控件

CustomValidator 是用于实现自定义验证的控件，支持客户端脚本验证和服务器验证两种方式。表 6-4 列出了 CustomValidator 控件的常用属性。

表 6-4 CustomValidator 控件的常用属性

属　　性	说　　明
ControlToValidate	要进行验证的控件 ID
ClientValidationFunction	设置客户端验证的脚本函数
OnServerValidate	服务器端验证的方法

6. ValidationSummary 控件

ValidationSummary 是用于汇总来自各个验证控件的错误报告的控件。如果不使用它，验证错误都直接显示在验证控件的位置。有时需要对显示的样式进行一些特定的控制，就需

要使用汇总控件将出错信息汇总在一个固定位置的列表中。表 6-5 列出了 ValidationSummary 控件的常用属性。

表 6-5　ValidationSummary 控件的常用属性

属　　性	说　　明
ShowMessageBox	设定是否显示弹出消息
ShowSummary	设定是否显示报告内容

注意事项

当 ValidationSummary 显示错误时，在验证控件的位置还是会显示出错误提示信息。可以将验证控件的 Text 属性设置为“*”，这样在显示错误提示时，验证控件的位置仅显示一个红色的“*”。另外，也可以不设置 Text 属性，而是直接在验证控件的标签中填写“*”，也可以获得同样的效果。

6.1.3　任务实施

1．设计用户注册页面

注册页面是用户用来填写一些相关的基本信息的页面，一般都是必填内容，所以需要为每一项都增加一个 RequiredFieldValidator 验证控件。

确认密码和密码一样，可以直接采用 CompareValidator 控件进行验证。设置 ControlToCompare 属性为用户输入密码控件的 ID，ControlToValidate 属性为输入确认密码控件的 ID。

电子邮件地址有固定的格式，可以使用 RegularExpressionValidator 控件进行验证，由于采用弹出窗口的方式来显示错误信息提示，因此需要设置 ValidationSummary 的 ShowMessageBox 属性为 True，ShowSummary 属性为 False。另外，设置每个验证控件的 Text 属性为“*”。

用户注册页面 UserTegister.aspx 的主要代码如下：

```
<table width="60%" height="22" border="0" align="center" cellpadding="0" cellspacing= "0">
  <tr>
    <td width="2" bgcolor="#DDDDCC"> </td>
    <td><div align="center">
    <table height="61" cellpadding="0" cellspacing="0">
        <tr>
            <td height="33" colspan="6"><p class="STYLE2">注册我的易购</p>
            </td>
        </tr>
          <tr>
            <td width="24%" align="center" valign="top" style="font-size: 11px;">用户名</td>
            <td valign="top" width="37%" align="left" style="height: 26px">
                <asp:TextBox ID="txtLoginId" runat="server"></asp:TextBox>
                <asp:RequiredFieldValidator ID="rfvTrueName" runat="server" ErrorMessage=
"请输入真实姓名" ControlToValidate="txtName">*
```

```
</asp:RequiredFieldValidator></td>
            </tr>
            <tr>
            <td width="24%" height="26" align="center" valign="top" style="font-size: 11px">真实姓名:
</td>
                  <td valign="top" width="37%" align="left">
                  <asp:TextBox ID="txtName" runat="server"></asp:TextBox>
<asp:RequiredFieldValidator ID="rfvUserName" runat="server"
ErrorMessage="请输入用户名" ControlToValidate="txtLoginId">*
</asp:RequiredFieldValidator></td>
            </tr>
            <tr>
            <td width="24%" height="26" align="center" valign="top" style=
            "font-size: 11px">密码: </td>
                  <td valign="top" width="37%" align="left">
                  <asp:TextBox ID="txtLoginPwd" runat="server" TextMode="Password" Width="155px">
</asp:TextBox>
                  <asp:RequiredFieldValidator ID="rfvPass" runat="server" ErrorMessage="请输入密
码" ControlToValidate="txtLoginPwd">*
</asp:RequiredFieldValidator></td>
            </tr>
      <tr>
            <td width="24%" height="26" align="center" valign="top" style=
            "font-size: 11px">确认密码:
            </td>
            <td valign="top" width="37%" align="left">
            <asp:TextBox ID="txtPwdAgain" runat="server" TextMode="Password" Width="155px">
</asp:TextBox> 
            <asp:CompareValidator ID="cvPass" runat="server" ErrorMessage="两次密码不一致"
ControlToCompare="txtLoginPwd" ControlToValidate=
            "txtPwdAgain">*</asp:CompareValidator>
            </td>
      </tr>
      <tr>
            <td width="24%" height="26" align="center" valign="top" style=
            "font-size: 11px">Email:
            </td>
            <td valign="top" width="37%" align="left">
            <asp:TextBox ID="txtEmail" runat="server"></asp:TextBox>
            <asp:RequiredFieldValidator ID="rfvEmail" runat="server" Error
            Message="请输入 Email" ControlToValidate="txtEmail">*
            </asp:RequiredFieldValidator>
            <asp:RegularExpressionValidator ID="revEmail" runat="server" ErrorMessage="格式错
误" ControlToValidate="txtEmail"
            ValidationExpression="\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*">*</asp:
RegularExpressionValidator>
            </td>
      </tr>
      <tr>
```

```
            <td width="24%" height="26" align="center" valign="top" style="font-size: 11px">地址：
            </td>
            <td valign="top" width="37%" align="left">
            <asp:TextBox ID="txtAddress" runat="server"></asp:TextBox>
            <asp:RequiredFieldValidator ID="rfvAddr" runat="server" Error
            Message="请输入地址" ControlToValidate="txtAddress">*
            </asp:RequiredFieldValidator> 
            </td>
        </tr>
        <tr>
            <td width="24%" height="26" align="center" valign="top" style="font-size: 11px">手机：</td>
            <td valign="top" width="37%" align="left">
            <asp:TextBox ID="txtTele" runat="server"></asp:TextBox>
            <asp:RequiredFieldValidator ID="rfvTel" runat="server" ErrorMessage="请输入手机号
" ControlToValidate="txtTele">*
            </asp:RequiredFieldValidator>
            </td>
        </tr>
        <tr>
            <td width="24%" height="26" align="center" valign="top" style="font-size: 11px">验证码：
            </td>
            <td valign="top" width="37%" align="left">
            <asp:TextBox ID="txtCode" runat="server"></asp:TextBox> 
            <cc1:SerialNumber ID="snCode" runat="server">
            </cc1:SerialNumber> 
            </td>
        </tr>
        <tr>
            <td colspan="2" align="left"><font color="red">
            <asp:ValidationSummary ID="vsregister" runat="server" Show
            MessageBox="True" ShowSummary="False" />
            </font>
            </td>
        </tr>
        <tr>
            <td colspan="2" align="center">
            <asp:ImageButton id="btnRegister" onclick="btnSubmit_Click" runat=
            "server" ImageUrl="../Images/az-finish.gif"></asp:ImageButton>
            <asp:Literal id="ltMain" runat="server"></asp:Literal>
            </td>
        </tr>
    </table>
    <div class="STYLE5">------------------------------</div>
    </div>
    </td>
    <td width="2" bgcolor="#DDDDCC"> </td>
</tr>
</table>
```

2．编写后台验证代码

有了验证控件来进行数据验证后，提交按钮的后台代码就比较简单了。不过在后台还需要判断用户填写的用户名是否已经被使用，若没有，则可以新建一个用户，然后向数据库中保存用户信息。单击事件的代码如下：

```
protected void btnSubmit_Click(object sender, ImageClickEventArgs e)
{
    User user = new User();
    user.LoginId = this.txtLoginId.Text;
    user.LoginPwd = this.txtLoginPwd.Text;
    user.Name = this.txtName.Text;
    user.Address = this.txtAddress.Text;
    user.Phone = this.txtTele.Text;
    user.Mail = this.txtEmail.Text;
    if (!UserManager.Register(user))
    {
        this.ltMain.Text = "<script>alert('用户名已使用！请重新选择！') </script>";
    }
        else
    {
        this.ltMain.Text = "<script>alert('注册成功！请继续购物'); window.location='../default.aspx'
</script>";
    }
}
```

问题思考

在 CustomValidator 中有两种不同的验证方式：一种是客户端脚本的验证；另一种是服务器端事件的验证。这两种验证方式各自有什么特点？

6.1.4 任务评价

标　　准	掌 握 情 况
能验证用户名不为空	熟练□ 基本□ 模糊□
能验证真实姓名不为空	熟练□ 基本□ 模糊□
能验证密码和确认密码不为空	熟练□ 基本□ 模糊□
能验证密码和确认密码需要相同	熟练□ 基本□ 模糊□
能验证 E-mail 不能为空而且符合电子邮件地址规则	熟练□ 基本□ 模糊□
能验证地址不能为空	熟练□ 基本□ 模糊□
能验证手机号码不能为空	熟练□ 基本□ 模糊□
自 我 小 结	

6.1.5　任务拓展

读者可在本节内容的基础上加强对两个部分的验证，具体要求如下：

- 真实姓名不能为空，而且至少为两个以上汉字。
- 手机号码必须要有 11 位。
- 直接在页面上显示提示信息，不要采用消息框来进行提示。

任务 6.2　修改商品信息

6.2.1　任务分析

任务目标：在实际使用中经常会碰到需要对商品信息进行补充和修改的情况。商品信息修改可以是批量修改也可以是独立修改。要实现批量修改通常需要在进行商品分类修改时通过多选的方式确定多个修改对象，然后通过一个特定的操作对这些选定对象进行一次性全部修改。但在本任务中，将实现商品信息的独立修改，即用户只需选定一个商品并查看该商品的详细信息，然后根据需要来修改其中的数据项即可。

完成标准：能够修改商品信息中各个数据项。

应用手段：通过定义 ObjectDataSource 的 Update 数据方法来实现数据的更新。

6.2.2　相关知识

6.2.2.1　ObjectDataSource

在 ObjectDataSource 中可以定义各种不同功能的数据方法，用来实现数据的查询、更新、添加和删除操作。在如图 6-4 所示的设置视图中可以看到 4 个选项卡，它们是分别用来设置不同类型的操作方法的。

图 6-4　设置 ObjectDataSource 的方法

本任务中要实现商品信息的修改，因此需要在 UPDATE 选项卡中指定用来实现修改的方法。当 ObjectDataSource 绑定的控件需要进行修改时，这个被指定的方法将会被自己调用，从而完成数据修改的操作。

6.2.2.2 日历控件

在数据的输入中，经常会碰到需要录入日期的情况，手动录入日期数据不仅比较麻烦，而且还容易在格式和内容上出错，因此，考虑采用日历控件来自动完成日期的录入。在 ASP.NET 中提供了 Calendar 控件来完成日期的自动录入，该控件的常用属性和事件如表 6-6 所示。

表 6-6 Calendar 控件的常用属性和事件

属性或事件	描 述
SelectedDate 属性	设置或获取日期
VisibleDate 属性	当前可见的日期
TitleFormat 属性	显示格式（“年月”或者“月”）
SelectionChanged 事件	一个日期被选中时触发的事件

还有另一类可用的日历控件叫做 JS 日历。JS 日历控件有很多种，通常采用 JavaScript 来实现，使用起来非常方便，在任务实施部分将会采用一个 JS 日历控件来实现日期的自动录入。

知识点小贴士

由于 JS 日历控件是采用 JavaScript 编写的，因此它们都具有无刷新的特点。与 ASP.NET 提供的 Calendar 控件不同，JS 日历控件在选择日期上不会造成数据的回发，而 Calendar 控件在每次日历的显示、隐藏和选择时都会发生数据回发。在实际使用中，JS 日历控件被采用得更多。

6.2.3 任务实施

1．编写逻辑层方法

在实现数据修改方法前，首先需要确认哪些数据是会被修改的，然后根据这些修改内容来编写相应的逻辑层方法。例如，在商品信息修改中，就需要修改商品名称、品牌、价格、发布日期、品牌、描述以及图片数据。基于这些要求，可以在逻辑层设计如下的商品信息修改方法：

```
public static void ModifyItem(string title, int brandId, decimal price, DateTime publishDate, string
description,  int id)
{
if (description == null)
description = "";
```

```
Item item = ItemService.GetItemById(id);                    //取出当前商品
item.Title = title;                                         //修改名称
item.Price = price;                                         //修改价格
item.Brand = BrandService.GetBrandById(brandId);            //修改品牌
item.PublishDate = publishDate;                             //修改发布日期
item.Description = description;                             //修改商品描述
ItemService.ModifyItem(item);                               //数据修改方法
}
```

2．配置数据源

修改数据的方法编写好后就可以在 ObjectDataSource 中设置用来进行修改的方法了，如图 6.4 所示。为了能够在 DetailView 控件中实现编辑功能，需要先在其智能标记中启用编辑，如图 6-5 所示。修改商品信息时，由于数据也是由用户从客户端进行输入的，需要对用户输入的数据进行验证，这时也需要使用到验证控件。但是在 DetailView 控件中并不能直接使用验证控件，先要把所有的列转换成模板列，然后再添加验证控件。其中，价格和日期的验证比较特殊，需要使用到正则表达式。

图 6-5　在 DetailView 中启用编辑

3．编写页面代码

另外一个特殊的地方是商品的品牌，品牌是已经存在于数据库中的内容，在修改品牌时也只能从限定的内容中选择，因此可以考虑使用下拉列表的形式。但是需要让下拉列表在编辑时才显示出来，所以在绑定事件中要对下拉列表的内容进行控制，代码如下：

```
protected void dvItemList_DataBound(object sender, EventArgs e)
{
    if (dvItemList.CurrentMode == DetailsViewMode.Edit)
    {
    DropDownList ddlBrand = this.dvItemList.FindControl("ddlBrand")
    as DropDownList;
        HiddenField hfBrand = this.dvItemList.FindControl("hfBrandId")
    as HiddenField;
        ddlBrand.SelectedValue = hfBrand.Value.Trim();
    }
}
```

当然下拉列表的值传递给更新方法还需要作一些处理，可以通过如下的代码实现：

```
protected void dvItemList_ItemUpdating(object sender, DetailsViewUpdateEvent Args e)
```

```
{
    //获得出版社下拉列表的值
    DropDownList ddlBrand = this.dvItemList.FindControl("ddlBrand") as DropDownList;
    //添加出版社 id 的参数
this.odsItems.UpdateParameters.Add("BrandId", ddlBrand.SelectedValue);
}
```

其中的 ItemUpdating 方法会在更新方法调用时发生，通过如上的这段代码，就可以在更新前将参数添加到参数列表中。完成后的页面主要代码如下：

```
//省略前面的代码
<asp:ObjectDataSource ID="odsItems" runat="server" SelectMethod="GetItemById"
TypeName="eBuyShop.BLL.ItemManager" InsertMethod="AddItem"UpdateMethod=
"ModifyItem" >
<SelectParameters>
<asp:QueryStringParameter DefaultValue="0" Name="id" QueryStringField="id" Type="Int32" />
</SelectParameters>
<UpdateParameters>
    <asp:Parameter Name="title" Type="String" />
    <asp:Parameter Name="Brandid" Type="Int32" />
    <asp:Parameter Name="Price" Type="Decimal" />
    <asp:Parameter Name="PublishDate" Type="DateTime" />
    <asp:Parameter Name="Description" Type="String" />
<asp:QueryStringParameter DefaultValue="0" Name="id" QueryStringField="id" Type="Int32" />
</UpdateParameters>
<InsertParameters>
    <asp:Parameter Name="title" Type="String" />
    <asp:Parameter Name="Categoryid" Type="Int32" />
    <asp:Parameter Name="Brandid" Type="Int32" />
    <asp:Parameter Name="Price" Type="Decimal" />
    <asp:Parameter Name="PublishDate" Type="DateTime" />
    <asp:Parameter Name="Description" Type="String" />
</InsertParameters>
</asp:ObjectDataSource>
<asp:ObjectDataSource ID="odsBrand" runat="server" SelectMethod="GetAllBrands"
TypeName="eBuyShop.BLL.BrandManager">
</asp:ObjectDataSource>
<asp:ObjectDataSource ID="odsCategory" runat="server" SelectMethod="
GetAllCategories" TypeName="eBuyShop.BLL.CategoryManager">
</asp:ObjectDataSource>
//省略后面的代码
```

4．实现日期自动录入

编写页面代码后，就可以在 DetailView 中实现商品信息的修改。在此基础上还可以进行一些细节上的处理，如日期输入的控制。在填写日期数据时，直接手动录入比较麻烦，可以考虑采用日历控件来实现日期的选择录入。基于 6.2.2.2 节所述的 JS 控件的优点，在这个地方也采用 JS 日历控件来实现日期的自动录入。这里介绍的是 My97DatePicker 日历

控件，该控件可以从网上免费下载。当然，通过学习这个日历控件的使用方法，读者也可以下载一些其他的 JS 日历控件来使用。

将下载的文件放入站点的一个目录中，如放入应用程序根目录的 My97DatePicker 文件夹中。在需要应用该控件时，可以在页面增加这样的一段代码：

```
<script language="javascript" type="text/javascript" src="../My97DatePicker/WdatePicker.js" charset=
"gb2312"></script>
```

然后修改用来输入日期的文本框的代码为：

```
<asp:TextBox ID="txtDate" runat="server" Text='<%# Bind("PublishDate") %>' CssClass="Wdate"
onFocus="new WdatePicker(this,'%Y-%M-%D',true,'default')" >
```

</asp:TextBox>使用的效果如图 6-6 所示。

图 6-6　JS 日历控件

5．实现图片上传

图片有别于普通的数据，它不是存储在数据库中的，图片的更新实际上是从客户端再上传一张图片到服务器，覆盖原来的图片。在 ASP.NET 中，有一个专门用于文件上传的控件 FileUpload，如图 6-7 所示。

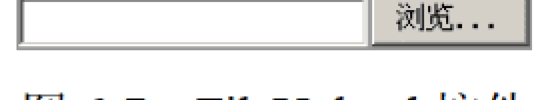

图 6-7　FileUpload 控件

有了 FileUpload 控件，用户就可以在 ItemUpdated 事件中编写代码实现图片的上传，代码如下：

```
protected void dvItemList_ItemUpdated(object sender, DetailsViewUpdatedEventArgs e)
{
    FileUpload fulItem = this.dvItemList.FindControl("fulItem") as FileUpload;
    Image imgItem = this.dvItemList.FindControl("imgItem") as Image;
    string FileName = fulItem.FileName;
    if (FileName.Trim().Length != 0)
    {
        string strpath = Server.MapPath(imgItem.ImageUrl);
        fulItem.PostedFile.SaveAs(strpath);                    //图片保存的路径
    }
}
```

6.2.4 任务评价

标　准	掌 握 情 况
能修改商品的名称	熟练□ 基本□ 模糊□
能修改商品的价格	熟练□ 基本□ 模糊□
能修改商品的品牌	熟练□ 基本□ 模糊□
能修改商品的描述	熟练□ 基本□ 模糊□
自 我 小 结	

6.2.5 任务拓展

在商品信息中有一项为商品图片，读者们可完善商品信息修改的功能，使其能实现商品图片的修改。提示：图片修改即采用新的图片对旧的图片进行覆盖。

本章小结

信息的添加和修改是数据处理中最常用到的两个方法。本章使用信息的添加来实现用户注册的功能，通过直接调用逻辑层中的方法来实现。而对于商品信息的修改则结合控件的使用来实现。其实对于商品信息的修改也可以采用调用逻辑层中的方法来实现，读者可以自己进行尝试。

自我检测

一、选择题

1．如果在信息输入中必须要填写身份证号码，则可以采用的验证控件是（　）。

A．RequiredFieldValidator 和 RangeValidator

B．CompareValidator 和 RequireFieldValidator

C．CompareValidator 和 RegularExpression

D．RequireFieldValidator 和 RegularExpression

2．ValidationSummary 控件的作用是（　）。

A．集中显示所有验证的结果　　B．使用验证控件时必须使用

C．验证求和的结果　　D．集合所有验证控件的功能

3．下列关于 ObjectDataSource 的说法中，正确的是（　）。

A．ObjectDataSource 控件使开发人员能够在保留他们的三层应用程序结构的同时，使用 ASP.NET 数据源控件

B．ObjectDataSource 控件使用反射创建业务对象的实例，并调用这些实例的方法以检索、更新、插入和删除数据

C．ObjectDataSource 控件不能接受参数

D．可以使用 ObjectDataSource 控件来编辑数据库数据

4．下列关于 DetailsView 控件的说法中，正确的是（　）。

A．DetailsView 控件能够直接对显示的数据进行分页

B．DetailsView 控件允许用户对数据库记录进行排序

C．DetailsView 控件可以逐一显示、编辑、插入或删除其关联数据源中的记录

D．DetailsView 一般用来显示详细信息

5．关于站点地图文件的说法中，正确的是（　）。

A．站点地图文件是一个 HTML 文件

B．站点地图文件中可以有多个 siteMap 标签

C．站点地图文件中可以有多个 siteMapNode 标签

D．站点地图文件反映的就是项目中页面文件的相对关系

6．The following statements on site map file, which are right（　）.

A．Site map file contains several siteMap labels

B．Site map file is a XML file

C．Site map file contains several siteMapNode labels

D．Site map file reflects the relationship between page files in project

二、简答题

1．可以通过 SQLDataSource、ObjectDataSource 等控件来实现数据库的访问，请简述它们之间的区别。

2．数据的验证可以在客户端完成也可以在服务器端完成，请简述这两种方式的区别以及它们各自的优缺点。

第 7 章　查 询 数 据

技能目标

1. 能使用 DataList 来实现列表显示。
2. 能使用 Repeater 完成精确列表显示。
3. 能编写代码实现分页。
4. 能编写代码实现排序。
5. 能实现数据的查询和显示。

相关词汇

英 文 单 词	中 文 含 义	英 文 单 词	中 文 含 义
query		current	
sort		bind	
authentication		demonstrate	
separator		state	

工作任务

任务 7.1　商品展示的实现
任务 7.2　查询商品信息
任务 7.3　发布商品信息 RSS

任务 7.1　商品展示的实现

7.1.1　任务分析

任务目标：Web 应用程序一般由前台和后台两部分组成。在第 6 章中所实现的如商品修改这些功能，事实上都是属于后台的内容，它需要有特定权限的用户才可以操作，对系统的普通用户是不可用的。而系统的前台是指可以被广大普通用户所访问的部分，在系统中，普通用户指的就是在这个购物系统中浏览商品和购买商品的用户，他们只需要能够查看到商品的信息并能够完成购买操作即可。在开发后台系统中，都是采用 GridView 来实现信息的展示，这种显示方法清晰简洁，而且方便进行更新、删除等操作，但是不方便进行

样式的设计。通常在前台系统中，普通用户对系统交互的要求比较高，因此，需要按照一定的样式将商品信息展示在普通用户的面前。

完成标准：按照一定的样式将商品信息进行展示，并能够提供排序和分页的功能。完成后效果如图 7-1 所示。

应用手段：采用 DataList 来实现商品的展示。

图 7-1 商品展示

7.1.2 相关知识

DataList 控件可用来显示限制于该控件的项目重复列表，通过编辑模板可多样化地显示数据。将一个 DataList 控件放入页面，可以看到如图 7-2 所示的效果。对其进行模板编辑时，可看到如图 7-3 所示的效果。

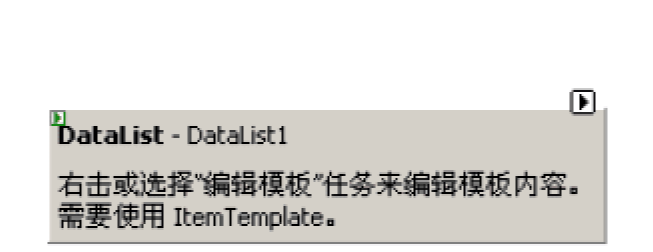

图 7-2 DataList 控件

图 7-3 模板编辑模式下的 DataList 控件

与 GridView 不同的是，DataList 控件不会显示出任何的样式或者表格，需要对它的模

板按照预先的设计进行编辑。打开模板编辑界面后，通过在模板中编辑 ItemTemplate 和 SeparatorTemplate 等项目来实现显示样式设计。通过在模板中包含 Web 服务器控件，可将列表项连接到代码，有了这些代码，用户可以在显示、选择和编辑模式之间进行切换。表 7-1 列出了 DataList 控件支持的模板，表 7-2 列出了 DataList 控件常用的属性。

表 7-1　DataList 控件的模板

模　　板	描　　述
ItemTemplate	数据源中的每行包含的 HTML 元素以及控件
AlternatingItemTemplate	在 DataList 控件中每隔一行呈现一次，通常用作为列表创建不同的外观，如对相邻的两行设置不同的背景颜色
SelectedItemTemplate	用户选中 DataList 控件中的项时呈现的元素。通常用于区别选取的项目与其他显示的项目
EditItemTemplate	处于编辑模式中时的布局。此模板通常包含可编辑的控件，如 TextBox 控件、DropDownList 控件等
HeaderTemplate	列表头部呈现的 HTML 元素以及控件
FooterTemplate	列表尾部呈现的 HTML 元素以及控件
SeparatorTemplate	每行之间呈现的元素。通常采用<hr>元素

表 7-2　DataList 控件常用属性

属　　性	描　　述
DataSource	设置或获取数据源
DataKeys	获取 DataKeyCollection，用作存储 DataList 控件中每行的键值
DataKeyField	设置或获取由 DataSource 属性指定的数据源中的键字段
SelectedIndex	设置或获取 DataList 控件中选定项的索引
SelectedItem	获取 DataList 控件中的选定项
Items	获取表示控件内单向的 DataListItem 对象的集合
RepeatColumns	设置或获取要在 DataList 控件中显示的列数
RepeatDirection	设置或获取 DataList 控件是水平显示或者垂直显示

知识点小贴士

通常情况下，美工人员会在静态页面上设计好数据展示的样式，开发人员只需要将显示内容贴入相应的模板，并把相关需要显示的内容换成绑定语句内容即可。

7.1.3　任务实施

1．商品信息展示模板的设计

首先需要在 DataList 的模板中设计好显示的样式，如图 7-4 所示。

图 7-4　DataList 数据显示样式

将其中的一些数据项进行数据绑定后就可以看到如图 7-4 所示的数据绑定提示，完成后的代码如下：

```
<asp:DataList ID="dlItems" runat="server">
<ItemTemplate>
        <table>
        <tr>
                <td rowspan="2">
                <a    href="ItemDetail.aspx?id=<%# Eval("Id")%>">
<img style="CURSOR: hand" height="94" alt="<%# Eval("Title") %>" src="<%# GetUrl(Eval("Id").ToString())
%>" width="125"    hspace="4"/></a>
                </td>
                <td style="FONT-SIZE: small; COLOR: red" width="650">
<a href="ItemDetail.aspx?id=<%# Eval("Id")%>" name="link_prd_name" target="_blank" class="booktitle"
id="link_prd_name"><%# Eval("Title") %></a>
                </td>
</tr>
<tr>
                  <td align="left">
                <br/>
                <span style="font-size:12px;line-height:20px;"><%# GetCut(Eval
("Description").ToString(),150) %></span>
                </td>
          </tr>
          <tr>
                  <td align="right" colspan="2">
<span style="font-size:13px;line-height:20px;font-weight:bold;"> &yen; <%# Eval("Price") %></span>
                </td>
          </tr>
          </table>
</ItemTemplate>
<SeparatorTemplate>
<hr />
```

```
</SeparatorTemplate>
</asp:DataList>
```

当然，到目前为止还没有实现商品信息的展示。从以上代码中可以发现，这里并没有采用 ObjectDataSource 直接进行数据源绑定。这里将结合数据排序和分页功能，采用另外的一种方法来进行数据的访问。

2．排序的实现

DataList 控件本身是没有排序以及分页功能的，需要开发人员自行设计后台代码来实现。实现排序最直接的方法是在数据库端将数据排好顺序，这样就可以在页面数据绑定时通过业务逻辑层传递相应的参数，然后在数据访问层访问数据时根据传递过来的参数直接添加排序条件。

为实现数据的排序，在数据访问层中设计一个带有参数的查询方法。此方法有两个参数：分类和排序条件。分类是用来区分商品的类别，排序条件是指定显示的商品按照什么条件排序，在本书项目里面商品可以按照发布日期和价格进行排序。因此，该查询方法的代码如下：

```
public static IList<Item> GetPartialItemsBySql(int categoryId, string order)
{
    string safeSql = "select Id, Title, BrandId, PublishDate, Price,SubString
     (Description,0,200) as ShortContent from items";
    if (categoryId > 0)
    {
    safeSql += " WHERE CategoryId = " + categoryId;
    }
    if (order.Trim().Length > 0)
    {
    safeSql += " order by " + order;
    }
return GetPartialItemsBySql(safeSql);
}
private static IList<Item> GetPartialItemsBySql(string safeSql)
{
    List<Item> list = new List<Item>();
    DataTable table = DBHelper.GetDataSet(safeSql);
    foreach (DataRow row in table.Rows)
    {
    Item item = new Item();
    item.Id = (int)row["Id"];
    item.Title = (string)row["Title"];
    if (table.Columns.Contains("Price"))
    item.Price = (decimal)row["Price"];
    if (table.Columns.Contains("ShortContent"))
    item.Description = (string)row["ShortContent"];
    if (table.Columns.Contains("PublishDate"))
    item.PublishDate = (DateTime)row["PublishDate"];
    if (table.Columns.Contains("CategoryId"))
```

```
        item.Category = CategoryService.GetCategoryById
         ((int)row["CategoryId"]);
        if (table.Columns.Contains("BrandId"))
        item.Brand = BrandService.GetBrandById((int)row["BrandId"]); //FK
                list.Add(item);
}
return list;
}
```

同时，也要在业务逻辑层中添加对应的方法来调用数据访问层中的查询方法，在业务逻辑层的 ItemManager 类中添加如下代码：

```
public static IList<Item> GetOrderedSmallItemsByCategoryId(int categoryId, string order)
{
    return ItemService.GetPartialItemsBySql(categoryId, order);
}
```

该方法也传递了一个分类参数 categoryId 和一个排序条件参数 order，而具体的排序功能还是在数据访问层中实现的。这样，在表示层进行数据绑定时，就可以通过两个参数的传递来获得相应的结果。

3．分页的实现

ASP.NET 中提供一种非常简便的分页方法：分页类 PagedDataSource。和 ObjectDataSource 一样，它也可以理解为一种具备特殊功能的数据源，该类封装了数据绑定控件以及和分页相关的数据，其常用的属性如表 7-3 所示。

表 7-3　PagedDataSource 类常用的属性

属　　性	描　　述
CurrentPageIndex	当前页数
PageCount	总页数
Count	总记录数
PageSize	每页要显示的记录数
DataSource	数据源
AllowPaging	设定是否实现自动分页

在使用时只要将数据源和当前页面数赋给 PagedDataSource 类的实例，其他的属性就可以自行计算得出。编写如下代码的绑定方法就可以比较方便地实现分页：

```
private void Databind()
{
    PagedDataSource pdsItems = new PagedDataSource();
    //对 PagedDataSource 对象的相关属性赋值
    pdsItems.DataSource = ItemManager.GetOrderedSmallItemsByCategoryId
    (Convert.ToInt32(ViewState["typeid"]), (string)ViewState["Order"]);
    pdsItems.AllowPaging = true;
    pdsItems.PageSize = 4;
```

```
        pdsItems.CurrentPageIndex = Pager;
lblCurrentPage.Text = "第  " + (pdsItems.CurrentPageIndex + 1).ToString() + "  页  共  " + pdsItems
.PageCount.ToString() + "  页";
        SetEnable(pdsItems);
        //把 PagedDataSource 对象赋给 DataList 控件
        dlItems.DataSource = pdsItems;
        dlItems.DataBind();
}
```

知识点小贴士

PagedDataSource 的使用一般可分为 3 步:

（1）指定数据源，在这里将 GetOrderedSmallItemsByCategoryId 方法返回的数据集作为数据源。

（2）分别设置 AllowPaging、PageSize 和 CurrentPageIndex 的属性。

（3）指定数据显示空间的数据源为该 PagedDataSource 实例。

由于有分页的存在，因此需要在不同的页面上保持如分页、分类和排序之类的信息，要采用一种方法将这些必须的参数保存下来以便能在不同的页面上访问。在 ASP.NET 中有很多对象可以用来保存数据，如 Session、Cookie 以及 Application 等，都可以用来保存数据。而在这里，由于这些分页和排序等参数只需要在该页面有效，因此可以直接采用 ViewState 来进行保存。ViewState 的用法和 Session 一样。

知识点小贴士

在 ViewState 中保存页数“1”可采用以下的写法：

```
ViewState["Page"]=1;
```

事实上，ViewState 状态保持的方式是在页面上放置一个隐藏域，每次回传数据的时候该隐藏域也一起回传，从而实现状态的保持。

现在可以编写完整的商品展示页面 ItemList.aspx，主要代码如下：

```
//省略前面的代码
<asp:DataList ID="dlItems" runat="server" OnSelectedIndexChanged
="dlItems_SelectedIndexChanged">
<ItemTemplate>
        <table>
        <tr>
        <td rowspan="2">
                <a    href="ItemDetail.aspx?id=<%# Eval("Id")%>">
<img style="CURSOR: hand" height="94" alt="<%# Eval("Title") %>" src="<%# GetUrl(Eval("Id").ToString())
%>" width="125"    hspace="4"/>
        </a>
        </td>
        <td style="FONT-SIZE: small; COLOR: red" width="650">
<a href="ItemDetail.aspx?id=<%# Eval("Id")%>" name="link_prd_name" target="_blank" class="booktitle"
```

```
id="link_prd_name">
        <%# Eval("Title") %></a>
                </td>
        </tr>
        <tr>
        <td align="left">
                    <br/>
                <span style="font-size:12px;line-height:20px;"><%# GetCut(Eval
          ("Description").ToString(),150) %></span>
                </td>
        </tr>
        <tr>
        <td align="right" colspan="2">
<span style="font-size:13px;line-height:20px;font-weight:bold;"> &yen; <%# Eval("Price") %></span>
                </td>
        </tr>
        </table>
        </ItemTemplate>
        <SeparatorTemplate>
        <hr/>
        </SeparatorTemplate>
</asp:DataList>
//省略后面的代码
```

相应的后台代码 ItemList.aspx.cs 如下：

```
protected void Page_Load(object sender, EventArgs e)
{
        if (!IsPostBack)
        {
                 //首次加载，赋初值
                 ViewState["Page"] = 0;
                ViewState["Order"] = "";
                try
                {
                          ViewState["typeid"] = Convert.ToInt32(Request.QueryString
          ["typeid"]);
                }
                catch
                {
                        ViewState["typeid"] = -1;
                    }
                Databind();
        }
}
private void Databind()
{
        PagedDataSource pdsItems = new PagedDataSource();
        //对 PagedDataSource  对象的相关属性赋值
        pdsItems.DataSource = ItemManager.GetOrderedSmallItemsByCategoryId(
```

```
        Convert.ToInt32(ViewState["typeid"]), (string)ViewState["Order"]);
        pdsItems.AllowPaging = true;
        pdsItems.PageSize = 4;
        pdsItems.CurrentPageIndex = Pager;
lblCurrentPage.Text = "第 " + (pdsItems.CurrentPageIndex + 1).ToString() + " 页 共 " + pdsItems.PageCount
.ToString() + " 页";
        SetEnable(pdsItems);
        //把 PagedDataSource 对象赋给 DataList 控件
        dlItems.DataSource = pdsItems;
        dlItems.DataBind();
}
public string GetUrl(string id)
{
        return StringHandler.CoverUrl(id);
}
public string GetCut(string content, int num)
{
        return StringHandler.CutString(content, num);
}
protected void btnDate_Click(object sender, EventArgs e)
{
        ViewState["Order"] = "PublishDate";
        Pager = 0;
        btnDate.Enabled = false;
        btnPrice.Enabled = true;
        Databind();
}
protected void btnPrice_Click(object sender, EventArgs e)
{
        ViewState["Order"] = "Price";
        Pager = 0;
        btnPrice.Enabled = false;
        btnDate.Enabled = true;
        Databind();
}
private void SetEnable(PagedDataSource pds)
{
        btnPrev.Enabled = true;
        btnNext.Enabled = true;
        if (pds.IsFirstPage)
        btnPrev.Enabled = false;
        if (pds.IsLastPage)
                btnNext.Enabled = false;
}
protected void btnNext_Click(object sender, EventArgs e)
{
        Pager++;
        Databind();
}
```

```
protected void btnPrev_Click(object sender, EventArgs e)
{
    Pager--;
    Databind();
}
/// <summary>
///  当前页数
/// </summary>
private int Pager
{
    get{return (int)ViewState["Page"];}
    set{ViewState["Page"] = value;}
}
```

7.1.4　任务评价

标　　准	掌 握 情 况
能显示商品信息	熟练□　基本□　模糊□
能分页显示商品信息	熟练□　基本□　模糊□
能对商品信息按照价格排序	熟练□　基本□　模糊□
能对商品信息按照发布日期排序	熟练□　基本□　模糊□
自 我 小 结	

7.1.5　任务拓展

在本节内容的基础上，读者可以进一步完善程序，使商品既可以按照价格从低到高排序，也可以按照价格从高到低排序。

任务 7.2　查询商品信息

7.2.1　任务分析

任务目标：查询是一项非常常用同时也是非常重要的技术，在绝大多数的系统中都有此功能。为方便用户快速找到合适的商品，在本书的项目中也提供了查询的功能。用户通过输入查询关键字来获得对应的查询结果，并将结果按照一定的格式进行显示。

完成标准：能够实现商品查询并按照一定的样式将商品信息进行展示，同时也能够提

供排序和分页的功能。完成效果如图 7-5 所示。

应用手段：采用 Repeater 来进行查询结果的展示。

图 7-5 查询结果显示

7.2.2 相关知识

Repeater 控件也可进行数据显示，它主要用于精确内容的显示，同样，它也需要编辑模板才可以使用，但是它在生成过程中不会产生任何多余的冗余布局代码。与 DataList 和 GridView 相比，Repeater 控件可用的模板更少，没有编辑和选择模板。因此必须在 Repeater 控件的模板内声明所有的 HTML 布局、格式以及样式。表 7-4 列出了 Repeater 控件支持的模板。

表 7-4 Repeater 控件的模板

模　板	描　述
ItemTemplate	定义列表中项目的内容和布局，为必选项
AlternatingItemTemplate	在 Repeater 控件中每隔一行呈现一次，通常用作为列表项创建不同的外观
HeaderTemplate	列表头部呈现的 HTML 元素以及控件
FooterTemplate	列表尾部呈现的 HTML 元素以及控件
SeparatorTemplate	每行之间呈现的用于分隔的元素

Repeater 控件是唯一允许开发人员在模板间嵌套 HTML 标记的控件。若要使用 Repeater 控件来创建表格则可以在 HeaderTemplate 中包含表格的开始标签<table>，在 ItemTemplate 中包含表格行的标记<tr>，在 FooterTemplate 中包含表格的结束标记</table>。

知识点小贴士

在放置 GridView 和 DataList 控件时，都将自动在页面中产生用于布局控制的如“<Table>”这样的标签。而 Repeater 的精确显示是指使用 Repeater 控件是不会产生任何的冗余代码，显示内容完全对应于开发人员在模板中设计的内容。

7.2.3　任务实施

1．查询功能的实现

用户可以通过页面上的搜索框来进行关键字搜索，如图 7-6 所示。

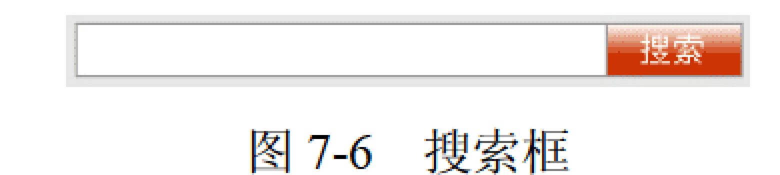

图 7-6　搜索框

搜索功能的实现其实和分类排序功能很类似，都需要在数据访问层中编写特定的方法来实现。参考任务 7.1 中的做法，在数据访问层的 ItemService 类中添加如下的代码：

```
public static IList<Item> SearchItems(string keyword, string order)
{
    string safeSql = "select Id, Title,BrandId, PublishDate,
    Price,SubString(Description,0,200) as ShortContent from Items";
    if (keyword.Trim().Length > 0)
    {
        safeSql += " where Title like '%" + keyword + "%'";
    }
    if (order.Trim().Length > 0)
    {
        safeSql += " order by " + order;
    }
    return GetPartialItemsBySql(safeSql);
}
```

通过这个方法就可以根据参数 keyword 来进行查询，同时还能够根据参数 order 来进行排列。相应地，也要在业务逻辑层的 ItemManager 中添加以下代码来进行参数的传递：

```
public static IList<Item> SearchItems(string keyword,string order)
{
    return ItemService.SearchItems(keyword,order);
}
```

这样，表示层就可以通过调用业务逻辑层中的 SerchItems 方法来进行参数传递并获取查询的结果。

2．查询结果的显示

查询页的做法也与任务 7.1 中商品展示的做法类似，不同的是查询结果显示页中用 Repeater 控件来显示查询的结果。同样，也需要编辑 Repeater 控件的模板来得到预定的显示样式，如图 7-7 所示。

图 7-7　使用 Repeater 来显示查询结果

查询结果显示页面 Search.aspx 的主要代码如下：

```
//省略前面的代码
<asp:Repeater ID="rpItemList" runat="server" OnItemCommand="rpItemList_
ItemCommand">
<ItemTemplate>
    <ul class="title_ul2">
    <li class="title_itemlist0" ><a href="ItemDetail.aspx?bid=<%# Eval("Id")%>">
    <%#Eval("Title") %></a></li>
    <li class="title_itemlist2"><%#Eval("Brand.Name")%> </li>
    <li class="title_itemlist3"><%#GetDate(Eval("PublishDate"))%> </li>
<li class="title_itemlist4"><%#GetMoney(Eval("Price"))%></li>
    </ul>
</ItemTemplate>
<AlternatingItemTemplate>
    <ul class="title_ul3">
    <li class="title_itemlist0" ><a href="ItemDetail.aspx?bid=<%# Eval("Id")%>">
    <%#Eval("Title") %></a></li>
    <li class="title_itemlist2"><%#Eval("Brand.Name")%> </li>
    <li class="title_itemlist3"><%#GetDate(Eval("PublishDate"))%> </li>
    <li class="title_itemlist4"><%#GetMoney(Eval("Price"))%></li>
    </ul>
</AlternatingItemTemplate>
</asp:Repeater>
//省略后面的代码
```

相应的后台代码 Search.aspx.cs 中的主要代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
        {
            //首次加载，赋初值
```

```
            ViewState["Page"] = 0;
            ViewState["Order"] = "";
            ViewState["KeyWord"]=Request.QueryString["KeyWord"];
            if (Request.QueryString["KeyWord"] != null && Request.QueryString
             ["KeyWord"].ToString().Trim() != "")
            {
                SearchKeywordManager.Search(Request.QueryString["KeyWord"]);
            }
            else
            {
                ViewState["KeyWord"] = "";
            }
            Databind();
        }
}
private void Databind()
{
        PagedDataSource pdsItems = new PagedDataSource();
        //对 PagedDataSource 对象的相关属性赋值
        pdsItems.DataSource = ItemManager.SearchItems(ViewState["KeyWord"].
        ToString(), ViewState["Order"].ToString());
        pdsItems.AllowPaging = true;
        pdsItems.PageSize = 20;
        pdsItems.CurrentPageIndex = Pager;
        lblCurrentPage.Text = "第" + (pdsItems.CurrentPageIndex + 1).ToString() + "页  共" + pdsItems
.PageCount.ToString() + "页";
        SetEnable(pdsItems);
        //把 PagedDataSource 对象赋给 DataList 控件
        rpItemList.DataSource = pdsItems;
        rpItemList.DataBind();
}
```

其中，分页和排序的方法可参考任务 7.1 中的代码。

7.2.4　任务评价

标　　准	掌 握 情 况
能按条件查询商品信息	熟练□　基本□　模糊□
能按要求显示查询结果	熟练□　基本□　模糊□
能对查询结果按发布日期排序	熟练□　基本□　模糊□
能对查询结果按价格排序	熟练□　基本□　模糊□
自 我 小 结	

7.2.5 任务拓展

根据查询的原理来完善查询的功能，读者可以自行完善查询功能，使得查询不仅仅可以根据商品名称来进行，还可以根据商品信息的其他字段来进行。

任务 7.3 发布商品信息 RSS

7.3.1 任务分析

任务目标：RSS 是目前信息发布中采用的一种主流技术，后台管理者可以通过 RSS 方式将指定的信息发布出来，而用户则可以通过 RSS 阅读器来订阅和阅读来自其他网站的信息。通过这种方式用户可以即时地获取最新的信息，并且不需要逐个访问感兴趣的站点。在这个任务中将要实现对商品信息的 RSS 发布，以便用户可以订阅和阅读来自站点的商品信息。

完成标准：能够将商品信息按照规范进行 RSS 发布，得到如图 7-8 所示的数据格式。

应用手段：采用 Repeater 控件来进行商品信息的 RSS 发布。

```
- <rss version="2.0">
  - <channel>
      <title>易购商城</title>
      <link />
      <description>易购商城</description>
      <copyright>Copyright 2008-2011 by 易购商城</copyright>
    - <item>
        <title>Samsung 三星  N148-DA06 10.1寸  上网本  黄色-N450 1G DDR2 250G 6芯</title>
        <description>时尚、精巧，是三星全新推出的N148系列上网本的代名词。它采用糖果色喷涂来突出可爱的气质，
          伤，有效保持崭新面貌。除了可爱的外表，N148还提供了丰富的功能性。ATOM处理器配合1GB内存，让网络
          电影；30万像素摄像头与麦克风、自带的三合一读卡器以及3个USB2.0接口让它的应</description>
        <link>http://localhost/web/ItemDetail.aspx?bid=6096</link>
        <pubDate>2010年7月6日</pubDate>
      </item>
    - <item>
        <title>SAMSUNG 三星  B5722C 2.8英寸  320万像素  双卡双待  GSM手机  灰褐色</title>
        <description>双卡一机，同时在线，双行其道，无须取舍  .</description>
        <link>http://localhost/web/ItemDetail.aspx?bid=6094</link>
        <pubDate>2010年5月5日</pubDate>
      </item>
```

图 7-8 商品信息 RSS 发布

7.3.2 相关知识

RSS 也叫聚合 RSS，是在线共享内容的一种简易方式（也叫聚合内容，Really Simple Syndication）。通常在时效性比较强的内容上使用 RSS 订阅能更快速地获取信息，网站提供 RSS 输出，有利于让用户获取最新更新的网站内容。实际上，RSS 发布就是提供一个 Feed 格式的文件，也就是说只需要按照 Feed 文件的格式来输出数据就可以实现信息的 RSS 发布。

知识点小贴士

Feed 文件实际上就是一个有规定格式的 xml 文件，只要按照这个格式生成 xml 文件就可以进行 RSS 发布。其格式如下：

```
<rss version="2.0">
<channel>
<title></title>
<link></link>
<description></description>
<copyright></copyright>
<item>
<title></title>
<description></description>
<link></link>
<pubDate></pubDate>
</item>
……
</channel>
</rss>
```

7.3.3 任务实施

1．获取需要发布的商品信息

在数据访问层的 ItemService 类中添加一个名为 GetNewItems 的方法，用来获取最新的 10 条图书信息，这 10 条最新的商品的信息将作为 RSS 的内容，其代码如下：

```
public static IList<Item>GetNewItems()
{
    return GetPartialItemsBySql("select Top 10 Id, Title,SubString
    (Description,0,200) as ShortContent, PublishDate FROM items order by PublishDate desc");
}
```

相应地，也需要在业务逻辑层的 ItemManager 类中添加 GetNewItems 方法来进行数据的传递，其代码如下：

```
public static IList<Item>GetNewItems()
{
    return ItemService.GetNewItems();
}
```

2．RSS 发布的实现

由于 Repeater 控件的特性，可以用它来实现 RSS 信息发布。首先新建一个名为 rss.aspx 的页面，将数据源控件和 Repeater 控件直接拖入页面即可。在这里没有涉及分页等功能，可以直接采用对象数据源控件。然后根据 RSS 的格式对 Repeater 控件的模板进行修改，页面代码如下：

```
<asp:Repeater id="rptRss" runat="server" datasourceid="odsNewItemsRss">
<HeaderTemplate>
<rss version="2.0">
    <channel>
    <title>易购商城</title>
    <link></link>
    <description>易购商城</description>
    <copyright>Copyright 2008-2011 by  易购商城</copyright>
</HeaderTemplate>
<ItemTemplate>
    <item>
    <title><%# EnCode(Eval("Title"))%></title>
    <description><%# EnCode(Eval("Description"))%></description>
    <link><%# EnCode(GetUrl(Eval("ID")))%></link>
    <pubDate><%# EnCode(string.Format("{0:D}", Eval("PublishDate")))%></pubDate>
    </item>
</ItemTemplate>
<FooterTemplate>
    </channel>
    </rss>
</FooterTemplate>
</asp:Repeater>
```

在绑定数据时采用了 EnCode 方法，由于在 RSS 格式的内容中不允许有 HTML 标签，因此需要采用 EnCode 方法对内容进行 HTML 编码，代码如下：

```
public string EnCode(object obj)
{
    return Server.HtmlEncode(obj as string);
}
public string GetUrl(object obj)
{
    return "http://"+Request.ServerVariables["HTTP_HOST"].ToString()+ "/
    web/ItemDetail.aspx?bid=" + obj as string;
}
```

通过 Repeater 控件，就可以非常方便地实现商品信息的 RSS 发布了。

知识点小贴士

RSS 是基于 XML 格式的，因此对格式有很严格的要求，尤其不能在内容中出现 HTML 标签。如果开发者不确定是否包含非法的字符，可以对他们进行 HTML 编码。

7.3.4 任务评价

标　准	掌 握 情 况
能理解 RSS 的工作原理	熟练□ 基本□ 模糊□
能使用 Repeater 控件生成符合 RSS 标准的 XML	熟练□ 基本□ 模糊□

续表

自 我 小 结

7.3.5　任务拓展

Atom 是另外的一种聚合技术，通过参照本节中 RSS 发布的方法，读者可以使用相关的控件来生成符合 Atom 标准的商品信息发布。

本章小结

DataList 和 Repeater 控件是进行数据显示的常用控件。在实际应用当中需要根据具体的情况来选择。这两个控件在使用时都需要通过模板的方式来设置其显示样式。其中 Repeater 控件专门用于内容的精确显示，不会生成任何冗余代码。与 Repeater 相比，DataList 提供了布局功能，会生成一个 HTML 的表格来容纳相关的信息。在如今以 DIV+CSS 为主流的页面布局中，DataList 有很大的局限性。

自我检测

一、选择题

1．在显示时不会自动增加额外标签的数据显示控件是（　）。

A．GridView 控件　　B．DataList 控件

C．Repeater 控件　　D．DetailsView 控件

2．设置换行样式的模板是（　）。

A．SeparatorTemplate　　B．ItemTemplate

C．TemplateField　　D．AlternatingItemTemplate

3．关于 Repeater 控件，下列说法不正确的是（　）。

A．Repeater 控件能够通过设置模板显示内容

B．Repeater 控件能够显示 HTML

C．Repeater 控件不会自动添加 HTML

D．Repeater 控件不会显示没有设置格式的内容

4．下列关于 PagedDataSource 的说法，正确的是（　）。

A．PagedDataSource 封装了数据绑定控件的分页功能

B．PagedDataSource 可以自动计算总页数、当前页数以及每页的显示条数

C．PagedDataSource 不能与 ObjectDataSource 同时使用

D．PagedDataSource 可以自动的实现分页和排序

5．下列关于 DataList 控件的说法，正确的是（　）。

A．DataList 控件可以替代 GridView 控件

B．DataList 控件内置了分页功能

C．DataList 控件可以套用自带的模板

D．DataList 只有模板列

6．In order to create RSS data format we can use a control to implement it, it is（　）.

A．DataList　　B．GridView

C．Repeater　　D．DetailsView

二、简答题

1．简述如何使用 PagedDataSource 来实现数据分页。

2．简述数据查询的原理。

3．Repeater 控件还能用于哪些场合？

第 8 章　完善页面功能

技能目标

1. 能使用验证码控件来制作验证码。
2. 能使用第三方控件进行富文本录入。

相关词汇

英 文 单 词	中 文 含 义	英 文 单 词	中 文 含 义
provide		assembly	
register		handle	
interaction		cute	
generate		serial	

工作任务

任务 8.1　实现验证码功能
任务 8.2　实现富文本输入

任务 8.1　实现验证码功能

8.1.1　任务分析

任务目标：验证码已被广泛地应用在 Web 系统的数据提交等位置，由于验证码当中包含的内容是随机产生的，每次数据提交时都需要输入不同的验证码，因此，验证码可以用来避免自动地提交数据。在这个任务中，直接采用验证码控件在用户注册页面上增加验证码功能。

完成标准：在用户注册页面上增加验证码功能，每次用户注册时都需要输入验证码，如图 8-1 所示。

应用手段：使用验证码控件 Webvalidates 来实现。

图 8-1　包含验证码的用户注册页面

8.1.2　相关知识

8.1.2.1　验证码

验证码的英文是 CAPTCHA，这个词最早是在 2002 年由卡内基梅隆大学的 Luis von Ahn、Manuel Blum、Nicholas J.Hopper 以及 IBM 的 John Langford 提出的。CAPTCHA 是“Completely Automated Public Turing test to tell Computers and Humans Apart”（全自动区分计算机和人类的图灵测试）的缩写，是一种区分用户是计算机还是人的公共全自动程序。它是由计算机生成问题并评判，但是必须只有人类才能解答。由于计算机无法解答 CAPTCHA 的问题，所以回答出问题的用户就可以被认为是人类。

验证码在页面上以图片的形式进行显示，当中包含有随机生成的文字、字母或者数字，一般通过程序绘制页面生成一幅图片并在其中加一些干扰像素。通过比对用户的输入和验证码中包含的数据来判断验证码的输入是否正确，从而认定一次数据提交是否有效。它能有效地防止针对某一个特定注册用户用特定程序暴力破解方式进行不断地登录尝试，实际上采用验证码是现在大多数网站通行的方式，也可以利用比较简易的方式实现验证功能。虽然登录比较麻烦，但是对系统访问者的密码安全来说，这个功能还是很有必要的。

常见的验证码的形式有如下的几种。

1．固定位数的数字和字母

用图片格式显示的固定位数的数字和字母，它可能全是数字或者全是字母，当然为了提高破解的难度，现在一般将数字和字母混合在一起使用。

2．随机显示的数字或字母

Google 采用了一种随机显示字母的验证，这种随机性体现在能够随机地控制显示字母的个数、随机地控制显示字母的字体以及随机地控制显示字母的颜色，具有更好的保护作用，其效果如图 8-2 所示。

图 8-2　Google 采用的随机验证码

3．汉字

现在也有一些网站采用汉字验证码的方式来进行验证，这样可以更进一步地提高安全性。不过这种做法有局限性，要求用户会读和打汉字，而且输入时略显烦琐。

4．问题验证码

问题验证码显示的验证码本身是一个提问，例如验证码显示的内容是：1+1=？，这个提问的内容本身也是随机生成的，用户需要回答出正确的答案才能完成验证。

8.1.2.2　验证码的实现

验证码都是以图片的形式显示在网页上，其中包含有随机生成的文字。在实现验证码时需要采用绘图的方式来进行，将随机生成的文字和一些干扰像素绘入一个指定大小的图片中，然后将图片显示在网页上。同时在后台需要保存验证码图片中的内容或者问题答案，用来与用户输入的验证码进行比对。

问题思考

大家会发现，在很多网站上的验证码控件图片中都会有很多杂乱的线条和点，用户很多时候都很难辨认验证码中的内容。这些线条和点称为干扰像素，那么这些干扰像素有什么作用呢？它们是采用什么方式来实现的呢？

8.1.2.3　验证码控件

验证码控件的作用是方便开发人员在页面上实现验证码功能，而不必自己去编写图形实现验证码。它们一般都提供了一些属性和方法来完成验证码的生成以及验证码的验证等功能。在 ASP.NET 中可以使用的第三方验证码控件也有很多，如 ValidateCode 以及下面即将要使用的 Webvalidates 控件等。

8.1.3　任务实施

1．添加 Webvalidates 控件

在这个任务中，直接使用验证码控件 Webvalidates 来实现验证码功能。Webvalidates 控件主要包含两个主要的方法 Create()和 CheckSN(String)，其中前者用来生成新的验证码，后者用来对比用户的输入。开发者在使用时，首先将 Webvalidates 控件添加到工具箱中，

然后再直接将 Webvalidates 控件放置到页面上即可。

添加了验证码控件的用户注册页面代码如下：

```
<%@ Register Assembly="WebValidates" Namespace="WebValidates" TagPrefix="cc1" %>
…（此处省略部分代码）
<tr>
    <td width="24%" height="26" align="center" valign="top" style="font-size: 11px">验证码：
    </td>
    <td valign="top" width="37%" align="left">
    <asp:TextBox ID="txtCode" runat="server"></asp:TextBox> 
    <cc1:SerialNumber ID="snCode" runat="server"></cc1:SerialNumber>  
    </td>
</tr>
```

2．实现验证码验证

验证码的生成和比对都可以通过编码实现，后台代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        snCode.Create();   //生成验证码
    }
}
protected void btnSubmit_Click(object sender, ImageClickEventArgs e)
{
    if (!CeckCode())
    {
        this.ltMain.Text = "<script>alert('验证码错误！ ')</script>";
        return;
    }
    User user = new User();
    user.LoginId = this.txtLoginId.Text;
    user.LoginPwd = this.txtLoginPwd.Text;
    user.Name = this.txtName.Text;
    user.Address = this.txtAddress.Text;
    user.Phone = this.txtTele.Text;
    user.Mail = this.txtEmail.Text;
    if (!UserManager.Register(user))
    {
        this.ltMain.Text = "<script>alert('用户名已使用！ 请重新选择！ ')</script>";
    }
    else
    {
        this.ltMain.Text = "<script>alert('注册成功！ 请继续购物');window.location='../default
        .aspx'</script>";
    }
}
protected bool CeckCode()
```

```
{
    if (snCode.CheckSN(txtCode.Text.Trim()))           //校验验证码
    {
        return true;
    }
    else
    {
        snCode.Create();
        return false;
    }
}
```

8.1.4　任务评价

标　　准	掌 握 情 况
能在开发环境中引用第三方控件	熟练□　基本□　模糊□
能使用验证码控件实现验证	熟练□　基本□　模糊□
能完成验证码“看不清”的更新	熟练□　基本□　模糊□
自 我 小 结	

8.1.5　任务拓展

结合验证码的实现原理，读者可以通过搜集资料，自行开发一个能在网页上生成验证码的类，并使用自己开发的类来实现验证码验证。

任务 8.2　实现富文本输入

8.2.1　任务分析

任务目标：对于一般比较简洁的数据可以直接使用文本框进行输入，但是对于有些包含格式或者其他信息的数据在输入时就需要采用富文本输入的方式。本任务就是通过使用富文本输入控件来实现商品信息的富文本输入。

完成标准：使用富文本控件来输入商品描述数据项，如图 8-3 所示。

应用手段：采用 FreeTextBox 来进行富文本输入。

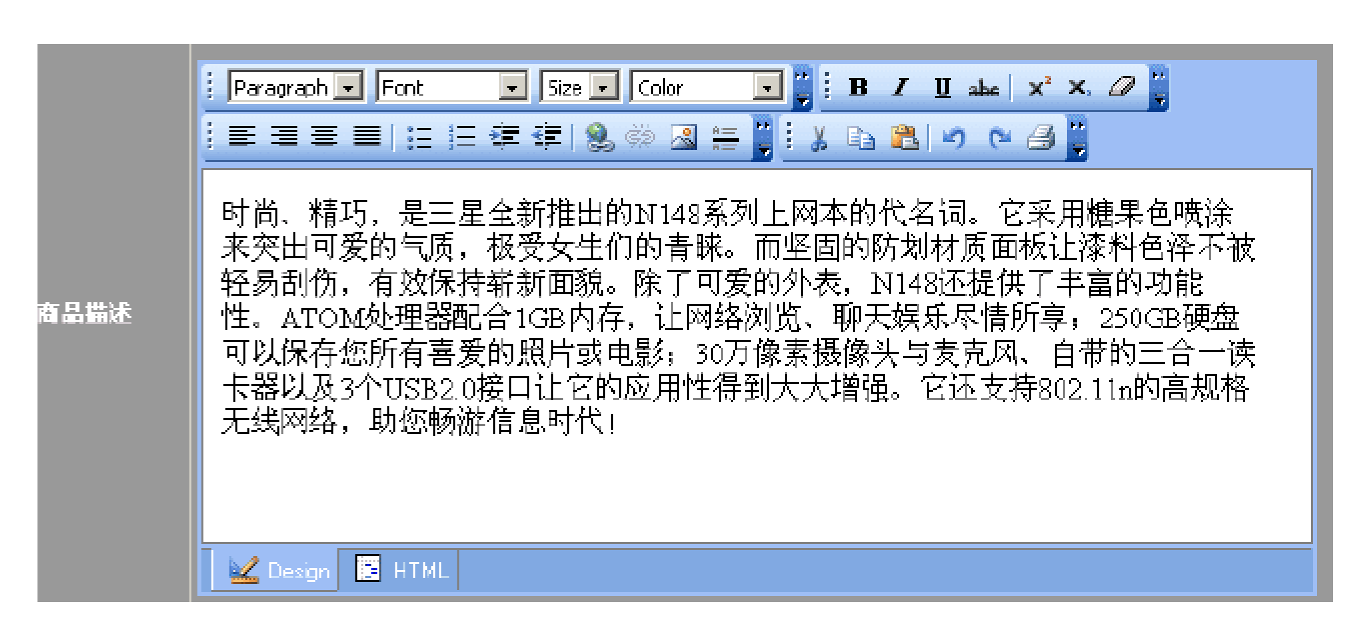

图 8-3　使用富文本输入商品描述

8.2.2　相关知识

8.2.2.1　富文本格式

富文本格式是由微软公司开发的跨平台文档格式。大多数的文字处理软件都能读取和保存 RTF 文档。

RTF 是 Rich Text Format 的缩写，意思是多文本格式。这是一种类似 DOC 格式（Word 文档）的文件，有很好的兼容性，使用 Windows 的“附件”中的“写字板”就能打开并进行编辑。RTF 是一种非常流行的文本格式，很多文字编辑器都支持它。一般的格式设置，例如，字体和段落设置，页面设置等信息都可以存在 RTF 格式中，它能在一定程度上实现 Word 与 WPS 文件之间的互访。这种规范是为了便于在应用程序之间轻松转储格式化文本和图形的一种编码方法。现在，用户可以利用特定转换软件，在不同系统如 MS-DOS、Windows、OS/2、Macintosh 和 Power Macintosh 的应用程序之间转换文本文档。RTF 规范提供的是一种在不同的输出设备、操作环境和操作系统之间交换文本和图形的一种格式。

知识点小贴士

RTF 文件的详细语法及关键字说明请参阅《Rich Text Format (RTF) Specification v1.7》。

8.2.2.2　富文本编辑控件

富文本编辑控件是一种在线的文本编辑控件，可以像 Word 编辑器一样输入内容，不需要编写 HTML 代码就能够直接设置样式、版式等。现在有很多非常好的富文本编辑控件可以采用，这些控件大致可以分为两类。

一类是与开发环境有关的控件，例如，在.NET 的环境中可以采用的富文本编辑器就有如下几种：

- RichTextBox：最早的富文本控件。
- CuteEditor：功能最完善的控件。
- FreeTextBox：使用简单方便的控件。
- FCKeditor：开源的富文本编辑器。

图 8-4 显示的是 CuteEditor 的输入界面。

还有一些 JavaScript 富文本编辑器，使用时也非常灵活，可以在页面上直接使用，如 MarkitUp、KingEditor 以及 EditArea 等。如图 8-5 所示的是 MarkitUp 的输入界面。

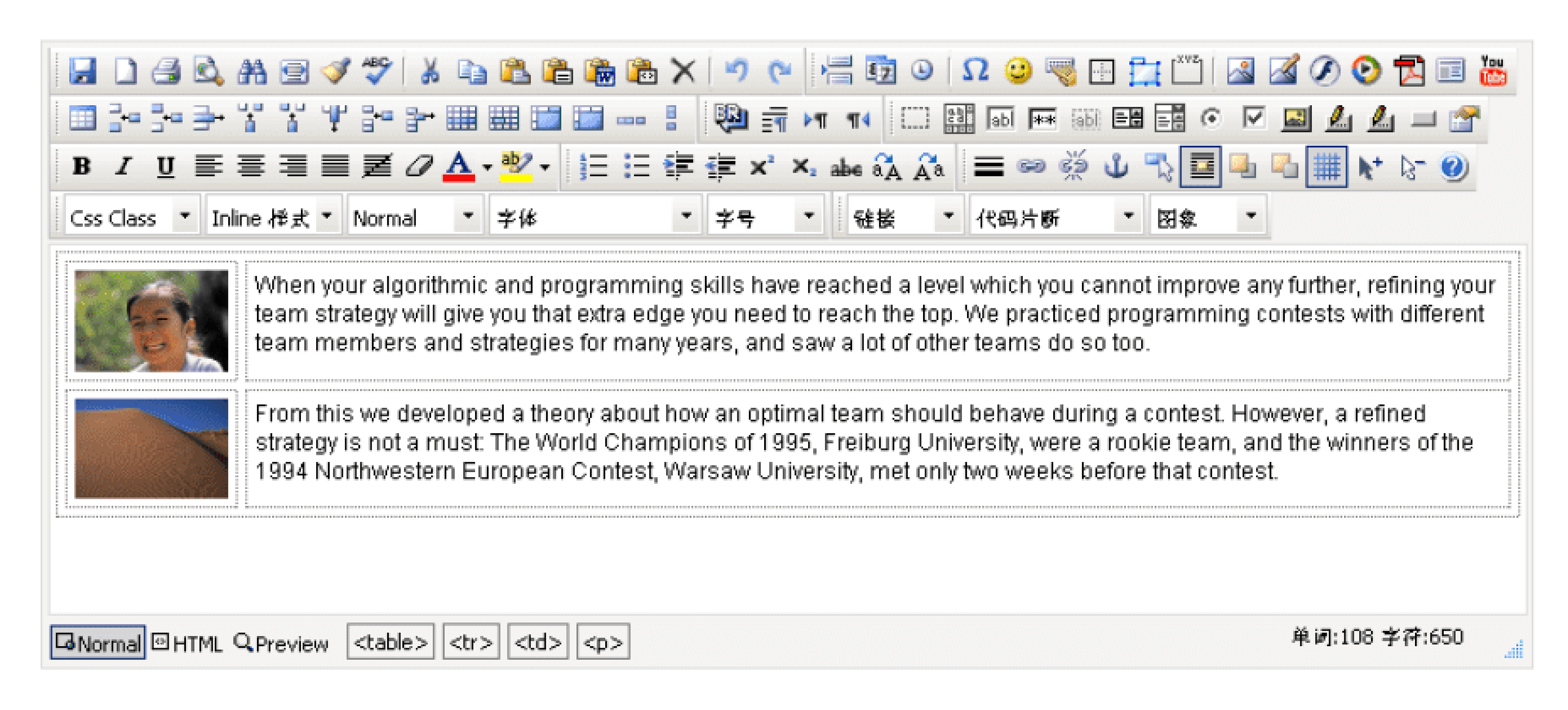

图 8-4　CuteEditor 富文本编辑器

另外，一些富文本编辑器可以用来在线编辑图片，如 AIE（Ajax Image Editor）控件，图 8-6 所示的是 AIE 的编辑界面。

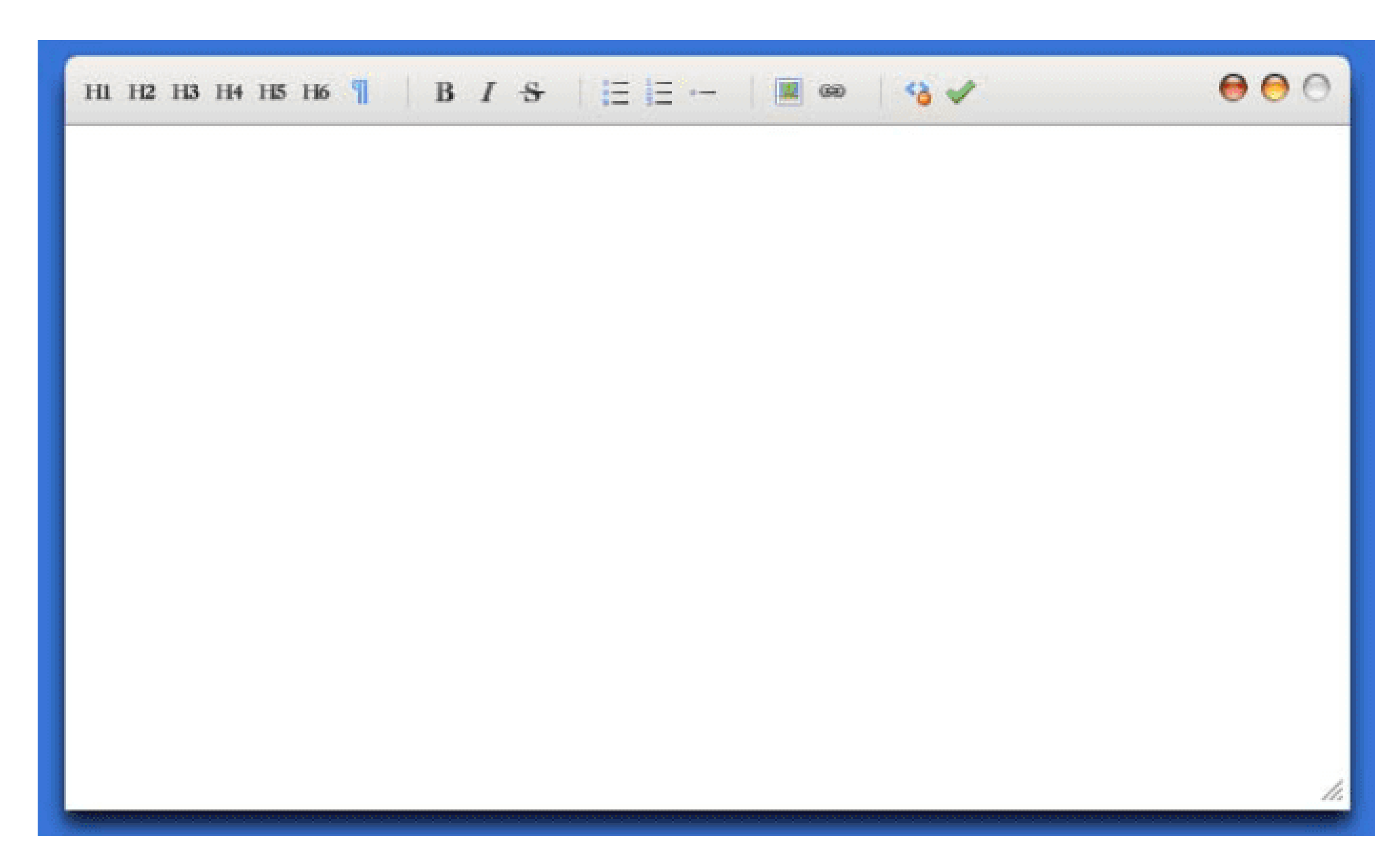

图 8-5　MarkitUp 富文本编辑器

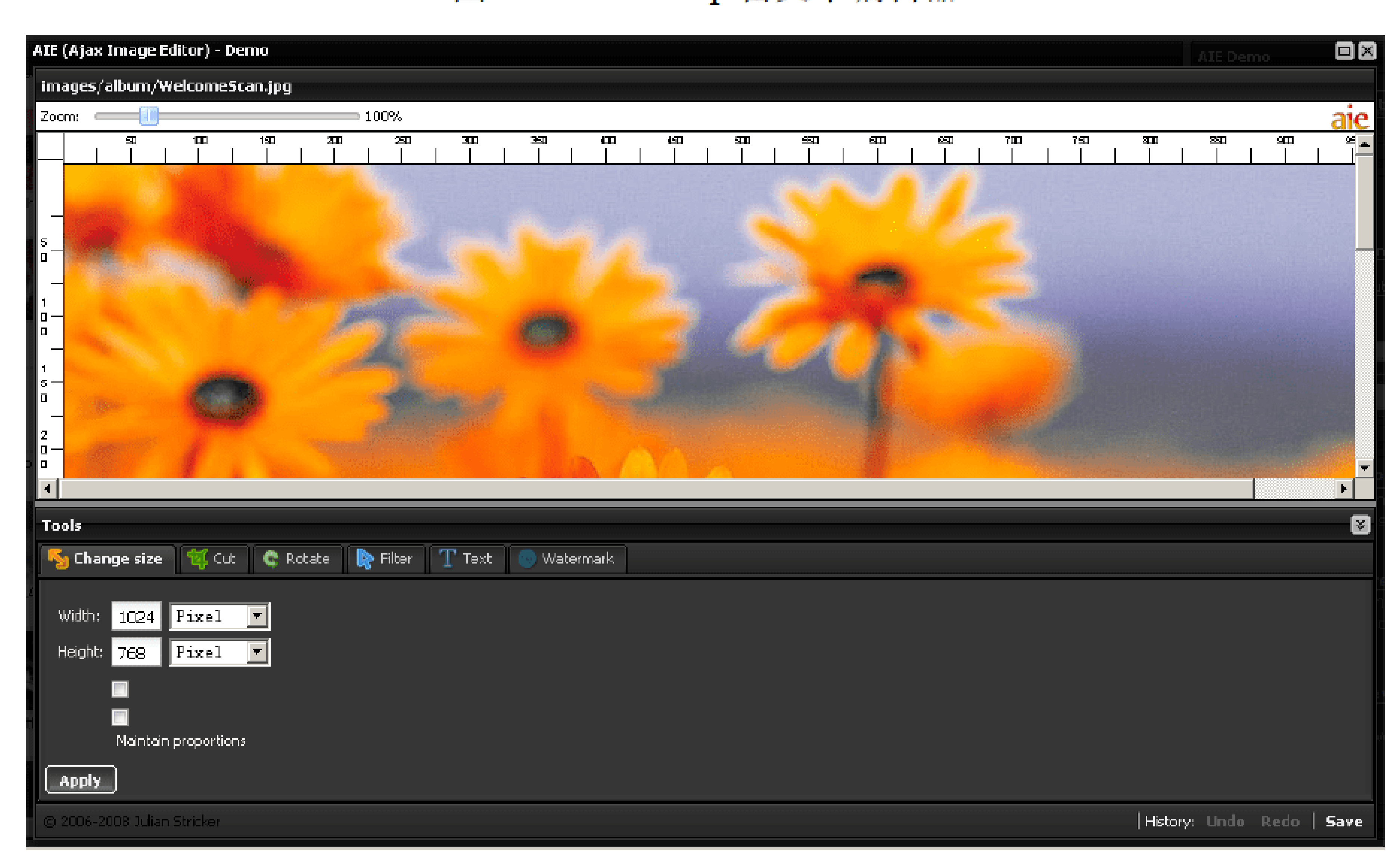

图 8-6　AIE 图片编辑器

问题思考

既然富文本编辑器有两种类型——控件形式和 JavaScript 形式的，那么大家可以思考一下在什么情况下采用控件形式？在什么情况下采用 JavaScript 形式呢？

8.2.3 任务实施

1. 导入 FreeTextBox 控件

在使用第三方控件前，都需要在开发环境中进行导入，导入的方法和步骤与 8.1.3 中导入 WebValidates 控件相同，读者可以自行完成。

2. 使用 FreeTextBox 控件

在这个任务中主要实现采用富文本的方式输入商品描述，因此需要将商品描述字段的编辑和添加模板换成 FreeTextBox 来实现。设计时，将编辑和添加模块自带的 TextBox 控件删除，并拖入 FreeTextBox 控件，同时将 Text 属性设置为绑定方法。具体代码如下：

```
<asp:TemplateField HeaderText="商品描述">
<EditItemTemplate>
    <FTB:FreeTextBox ID="FreeTextBox1" runat="server" Height="200px" Text='<%# Bind
("Description") %>'>
    </FTB:FreeTextBox>
</EditItemTemplate>
<InsertItemTemplate>
    <FTB:FreeTextBox ID="FreeTextBox1" runat="server" Height="200px" Text='<%# Bind
("Description") %>'>
    </FTB:FreeTextBox>
</InsertItemTemplate>
<ItemTemplate>
    <asp:Label ID="lbDesc" runat="server" Text='<%# Bind("Description") %>'></asp:Label>
</ItemTemplate>
</asp:TemplateField>
```

运行后，就可以得到如图 8-3 所示的效果了。FreeTextBox 提供 Design 和 HTML 两种不同的编辑视图，现在就可以通过 HTML 视图来查看 FreeTextBox 中内容的 HTML 编码。

注意事项

有时，编辑过程中会出现“检测到有潜在危险的 Request.Form 值”的错误。这是由于 ASP.NET 自身的安全机制引起的，它屏蔽了有潜在危险的表单提交，可以通过设置 Page 指令中的 ValidateRequest="false"来解决此问题。

8.2.4 任务评价

标　准	掌 握 情 况
能导入富文本编辑器控件	熟练□ 基本□ 模糊□
能在商品修改中使用富文本输入	熟练□ 基本□ 模糊□
能在商品添加中使用富文本输入	熟练□ 基本□ 模糊□

续表

自 我 小 结

8.2.5　任务拓展

读者可以选择一款 JavaScript 的富文本编辑器来实现富文本输入，并与使用 FreeTextBox 控件实现的富文本输入进行比较。

本章小结

本章主要阐述了两种用于特定功能的第三方控件的使用，用于验证码功能的 Webvalidates 控件以及用于富文本输入的 FreeTextBox 控件。通过这两个控件的使用，读者也可以了解在 Visual Studio 开发环境中引入第三方控件的方法。同时通过采用第三方控件的这种开发方法，读者也可以认识到在实际开发中为了提高开发的效率和质量，有时有必要购入或者采用第三方控件来解决问题，而并不是每个环节都需要从头开始。

自我检测

一、选择题

1．使用 Webvalidates 控件实现验证码功能时，首先调用的方法是（　）。

A．New()　　B．Create()
C．Init()　　D．Draw()

2．下列对于第三方控件的说法，正确的是（　）。

A．在系统开发时，不应该采用第三方提供的控件
B．第三方控件是由微软提供的没有包含在 VisualStudio 开发环境中的控件
C．第三方控件都是免费的，可以直接拿来使用
D．采用第三方控件是一种可以提高开发效率的方法

3．在使用 FreeTextBox 富文本编辑控件时，用来获取或设置文本内容的属性是（　）。

A．Value　　B．InnerText
C．Text　　D．InnerHtml

4．下列关于验证码的说法，不正确的是（　）。

A．验证码可用来防止暴力破解
B．验证码都是由数字组成的
C．验证码都是以图片的形式显示在网页中
D．验证码可以对用户输入的数据进行加密

5．下列关于 FreeTextBox 控件的说法，不正确的是（　）。

A．FreeTextBox 是一个 ASP.NET 开源服务器控件

B．FreeTextBox 是基于 JavaScript 技术的控件

C．FreeTextBox 可用来对图片进行编辑

D．FreeTextBox 控件是由微软开发的

6．The following statements on validate code, which is wrong?（　）

A．Validate code can be used to prevent brute force attack

B．Validate code is composed of number

C．Validate code is displayed as image on web page

D．Validate code can be used to encrypt the data user inputted

二、简答题

1．简述在 Visual Studio 2005 开发环境中导入第三方控件的方法。

2．在哪些情况下可以采用第三方控件进行开发？

第 9 章　完善系统功能

技能目标

1．能使用 HttpHandler 实现图片水印。
2．能通过 Web Service 发布商品查询功能。

相关词汇

英 文 单 词	中 文 含 义	英 文 单 词	中 文 含 义
handler		reusable	
watermark		physical path	
filter		rectangle	
context		SOAP	

工作任务

任务 9.1　添加图片水印
任务 9.2　发布商品查询 Web Service

任务 9.1　添加图片水印

9.1.1　任务分析

任务目标：给商品图片加上数字水印是目前绝大多数电子商务类网站通行的做法，其目的是为了避免自己的商品图片被任意地转载和使用，具有很好地保护自身图片所有权的作用。在本任务中，将采用 ASP.NET 的 HTTP 请求机制来实现自动为商品图片添加水印的功能，使商品图片在显示时能自动的打上预先设计好的数字水印。

完成标准：在商品管理页面的编辑和修改功能中提供水印功能，使用户在上传商品图片的时候能自动完成图片水印的添加，其效果如图 9-1 所示。

图 9-1　添加了水印的商品图片

应用手段：使用 HTTP 请求机制实现。

9.1.2 相关知识

9.1.2.1 数字水印

数字水印是向多媒体数据（如图像、声音、视频信号等）中添加某些数字信息以达到文件真伪鉴别、版权保护等目的。嵌入的水印信息隐藏于宿主文件中，不影响原始文件的可观性和完整性。

数字水印过程就是在被保护的数字对象（如静止图像、视频和音频等）中嵌入的某些能证明版权归属或跟踪侵权行为的信息，可以是作者的序列号、公司标志、有意义的文本等。与水印相近或关系密切的概念有很多，从目前出现的文献中看，已经有信息隐藏（Information Hiding）、信息伪装（Steganography）、数字水印（Digital Watermarking）和数字指纹（Fingerprinting）等概念。数字水印应有如下基本特征。

1．可证明性

水印应能为受到版权保护的信息产品的归属提供完全和可靠的证据。被嵌入到保护对象中的所有者的有关信息（如注册的用户号码、产品标志或有意义的文字等）并能在需要的时候将其提取出来。水印可以用来判别对象是否受到保护，并能够监视被保护数据的传播、真伪鉴别以及非法复制控制等。这实际上是发展水印技术的基本动力，虽然从目前的文献来看，对其研究相对少一些，就目前已经出现的很多算法而言，攻击者完全可以破坏掉图像中的水印，或复制出一个理论上存在的“原始图像”，这导致文件所有者不能令人信服地提供版权归属的有效证据。因此一个好的水印算法应该能够提供完全没有争议的版权证明，在这方面还需要做很多工作。

2．不可感知性

不可感知包含两方面的意思，一个指视觉上的不可见性，即因嵌入水印导致图像的变化对观察者的视觉系统来讲应该是不可察觉的，最理想的情况是水印图像与原始图像在视觉上一模一样，这是绝大多数水印算法所应达到的要求；另一方面，水印用统计方法也是不能恢复的，例如，对大量的用同样方法和水印处理过的信息产品，即使用统计方法也无法提取水印或确定水印的存在。

3．鲁棒性

鲁棒性问题对水印而言极为重要。鲁棒性是一个技术术语，简单地说，就是指一个数字水印应该能够承受大量的、不同的物理和几何失真，包括有意的（如恶意攻击）或无意的（如图像压缩、扫描与复印、噪声污染、尺寸变化等）。在经过这些操作后，鲁棒的水印算法应仍能从水印图像中提取出嵌入的水印或证明水印的存在。如果不掌握水印的所有有关知识，数据产品的版权保护标志应该很难被伪造。若攻击者试图删除水印则将导致多媒体产品彻底破坏。假设一个读者在网上下载了数字图书馆发布的作品，打印出来并非法大量散发以牟取利益，那么包含水印的作品应能在有物理失真的情况下依然提供足够的版权证据。

如图 9-2 所示为新蛋商城和京东商城上使用的添加了数字水印的商品图片。

图 9-2　新蛋商城和京东商城的商品图片

知识点小贴士

其实这里所说的图片数字水印和标准意义的数字水印的概念是有区别的，图片上附加的数字水印主要是起到标示和版权说明的作用，以防止图片被他人直接使用，因此与标准意义上具有不可感知性和鲁棒性的数字水印有所区别。

要在商品图片上添加水印信息可以采用如下的几种方式。

1．编辑每张商品图片

在处理商品图片时，直接通过图片处理工具将水印图片添加到商品图片上。如果图片量很小，采用这种方法是可行的。但是，如果图片的量很大，采用这种方法将会耗费大量的人力和时间。虽然有些图片处理工具有批量处理的功能，但这种方式还有一个致命的缺点就是将水印直接加到原始图片上会破坏原来图片的视觉效果。

2．动态添加图片水印

动态添加图片水印是一种较为理想的添加水印的方法，一方面它的效率高，添加水印的过程可由代码直接完成；另一方面，它不会破坏原始图片的视觉效果，只是在页面上显示图片时将水印加在图片的上面，不会修改原始图片。在这个任务中将采用 HttpHandler 的方式来实现该功能。

9.1.2.2　Http 请求处理

当服务器端接收到对一个*.aspx 页面文件的 HTTP 请求时，这个请求会被 inetinfo.exe 进程截获，它在判断文件的后缀（.aspx）之后，将这个请求转交给 ASPNET_ISAPI.dll，ASPNET_ISAPI.dll 会通过 http 管道（Http PipeLine）将请求发送给 ASPNET_WP.exe 进程，在 ASPNET_WP.exe 进程中通过 HttpRuntime 来处理这个请求，处理完毕后将结果返回客户端。其中，inetinfo.exe 进程是 www 服务的进程，IIS 服务和 ASPNET_ISAPI.DLL 都寄存在此进程中。而 ASPNET_ISAPI.DLL 是处理.aspx 文件的 win32 组件。其实 IIS 服务器是只能识别.html 文件的，当 IIS 服务器发现被请求的文件是.aspx 文件时，IIS 服务器将其交给 aspnet_isapi.dll 来处理。aspnet_wp.exe 进程则是 ASP.NET 框架进程，提供.NET 运行的托管环境，.NET 的 CLR（公共语言运行时）就是寄存在此进程中。如图 9-3 所示是.NET Framework 处理一个 HTTP 请求的流程：

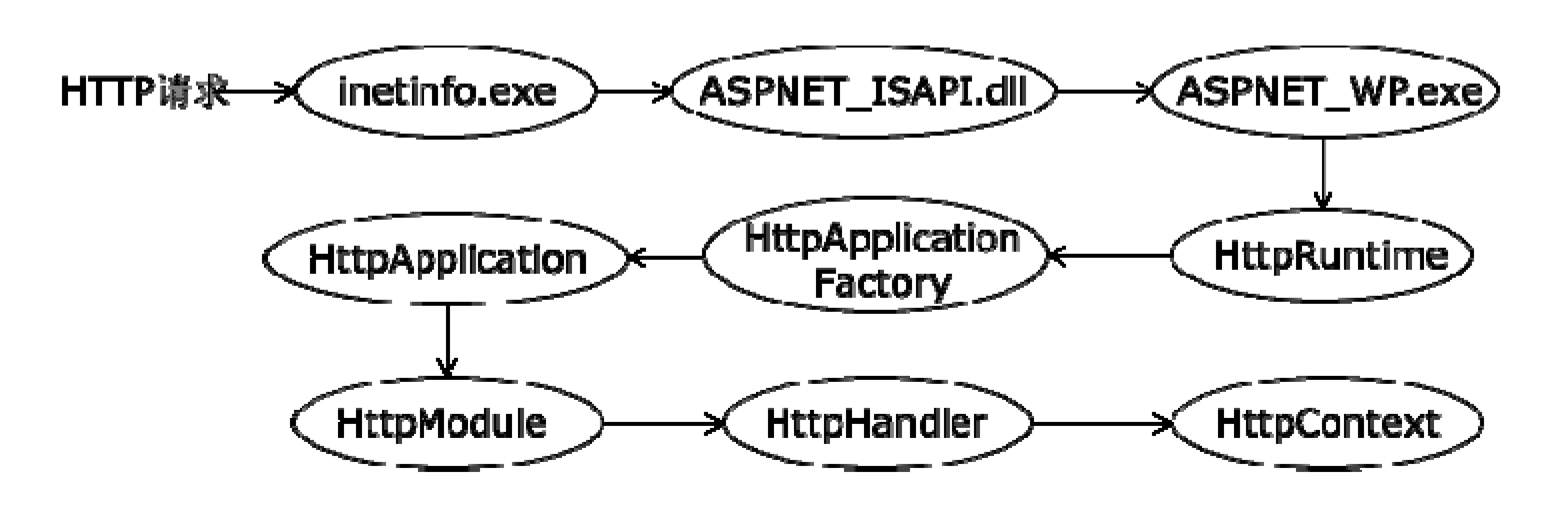

图 9-3　HTTP 请求的处理流程

ASP.NET 请求处理过程是基于管道模型的，这个管道模型是由多个 HttpModule 和 HttpHandler 组成，ASP.NET 把 HTTP 请求依次传递给管道中各个 HttpModule，最终被 HttpHandler 处理，处理完成后，再次经过管道中的 HTTP 模块，把结果返回给客户端。在每个 HttpModule 中都可以干预请求的处理过程的。

当请求到达 HttpModule 时，系统还没有对这个请求进行处理，但是可以在这个请求传递到处理中心（HttpHandler）之前附加一些其他信息，或者将截获的这个请求并作一些额外的工作，也或者终止请求等。在 HttpHandler 处理完请求之后，可以再在相应的 HttpModule 中把请求处理的结果进行再次加工返回客户端。因此，HttpHandler 才是 HTTP 请求的处理核心。

注意事项

在 HTTP 请求的处理过程中，只能调用一个 HttpHandler，但可以调用多个 HttpModule。

9.1.2.3　HttpHandler 开发

在 Visual Studio 2005 中，可以很方便地对 HttpHandler 进行开发。在“添加新项”对话框中通过直接添加“一般处理程序”来创建 HttpHandler 应用，如图 9-4 所示。

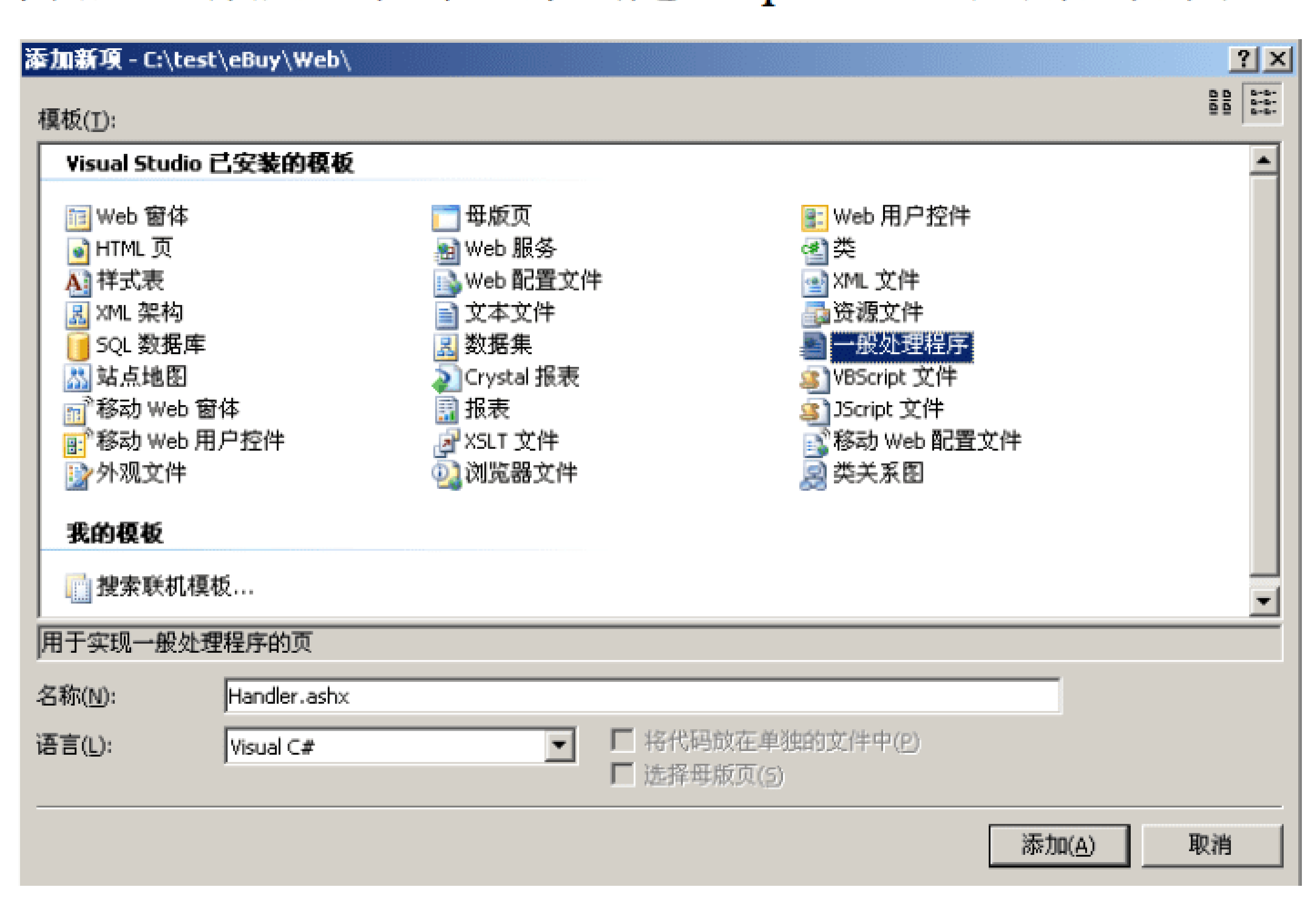

图 9-4　创建 HttpHandler 应用

HttpHandler 处理程序以.ashx 为后缀名，创建后的默认内容如下：

```
<%@ WebHandler Language="C#" Class="Handler" %>
using System;
```

```
using System.Web;
public class Handler : IHttpHandler {
    public void ProcessRequest (HttpContext context)
    {
        context.Response.ContentType = "text/plain";
        context.Response.Write("Hello World");
    }
    public bool IsReusable
    {
        get {return false;}
    }
}
```

其中，ContentType 可用于设置程序的输出类型，context 对象用于控制输出的内容。该默认程序可在被访问时显示“Hello World”的文字。需要使用 HttpHandler 处理的功能可写在 ProcessRequest 方法中，该方法有一个 HttpContext 类型的参数，它封装了有关 HTTP 请求的所有 HTTP 特定的信息，在这里，它被用在不同的 HttpModule 和 HttpHandler 之间传递数据，也可以用于保持某个完整请求的相应信息。

注意事项

实现 HttpHandler 的功能需要实现 IHttpHandler 接口，实现该接口则需要实现 IsReusable 属性和 ProcessRequest 方法。其中，IsReusable 属性用于设置是否可重用该 HttpHandler 的实例，而 ProcessRequest 方法则是整个 HTTP 请求的最终处理方法。

9.1.3　任务实施

1．创建 HttpHandler 程序

在这个任务中将要使用 HttpHandler 来对被请求的图片进行添加水印处理，为了达到这个目的，需要根据用户请求来找到对应的商品图片，然后使用图像处理的方法将事先准备好的水印图片加到商品图片的指定位置，最后，将添加了水印的商品图片再显示出来。如图 9-5 所示是事先准备好的水印图片。

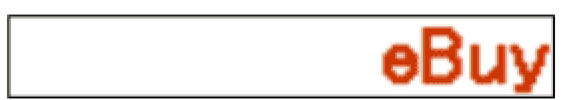

图 9-5　用作水印的图片

在工程中添加用于生成水印的 HttpHandler 处理程序 ItemImage.ashx，其代码如下：

```
<%@ WebHandler Language="C#" Class="ItemImage"   %>
using System;
using System.Web;
using System.Drawing;
using System.IO;
public class ItemImage : IHttpHandler {
    private const string COVERSADDR = "~/Images/Items/";              //保存商品图片的文件夹
    private const string WATERMARKADDR = "~/Images/watermark.gif";    //数字水印图片
```

```
    private const string DEFAULTIMAGE = "~/Images/default.jpg";          //默认图片

    public void ProcessRequest (HttpContext context)
    {
        string path = context.Request.MapPath(COVERSADDR + context.Request.
        Params["id"].ToString()+".jpg");//根据传递的商品编号获取的商品图片路径
        System.Drawing.Image image;
        if (File.Exists(path))
        {
            image = Image.FromFile(path);/                        /加载商品图片
            Image watermark = Image.FromFile(context.Request.MapPath
         (WATERMARKADDR));                                       //加载水印图片
            Graphics g = Graphics.FromImage(image);
            g.DrawImage(watermark, new Rectangle(image.Width-watermark.Width, image.Height-
watermark.Height, watermark.Width, watermark.Height), 0, 0, watermark.Width, watermark.Height,
GraphicsUnit.Pixel);                                             //在 image 上绘制水印
            g.Dispose();
            watermark.Dispose();                                 //释放水印图片
        }
        else
        {
            image = Image.FromFile(context.Request.MapPath(DEFAULTIMAGE));  //加载默认图片
        }
        context.Response.ContentType = "image/jpeg";             //设置输出格式
        image.Save(context.Response.OutputStream, System.Drawing.Imaging.
        ImageFormat.Jpeg);                                       //将图片存入输出流
        image.Dispose();
        context.Response.End();
    }
    public bool IsReusable
    {
        get {return false;}
    }
}
```

上述代码中最关键的实现部分是 Graphics 类的 DrawImage，它实现了将水印图片绘制在商品图片上的功能，绘制好的图片将以流的方式进行输出。

2．获取添加水印后的图片

通过 HttpHandler 已经可以实现对被请求的图片进行添加水印操作，那么处理好的图片该如何获取呢？例如，要获取编号为 435 的商品添加了水印的商品图片，可以直接访问 ItemImage.ashx?id=435 来获取，HttpHandler 会根据请求的内容来完成图片的处理并输出加了水印的图片。现在要做的事情就是修改显示模板中商品图片的显示方式，将商品图片的路径改为对应的 HttpHandler 请求路径，可对图片标签进行如下的修改：

```
<img alt="<%# Eval("Title") %>" src="ItemImage.ashx?id=<%# Eval("Id") %>" width="125" height="94" />
```

9.1.4　任务评价

标　　准	掌 握 情 况
能理解 HttpHandler 的工作原理	熟练□ 基本□ 模糊□
能进行 HttpHandler 的开发	熟练□ 基本□ 模糊□
能使用 HttpHandler 来实现图片水印的显示	熟练□ 基本□ 模糊□
自 我 小 结	

9.1.5　任务拓展

参照本节的内容，读者可以通过使用 HttpHandler，对已经没有库存的商品在显示时将其商品图片打上“缺货”的水印。

任务 9.2　发布商品查询 Web Service

9.2.1　任务分析

任务目标：为了能与其他网站信息互通以及方便其他网站能够获取本网站的商品信息，可以对外开发商品查询功能。对方网站可以通过开放的商品查询功能从本站点获取相应的商品信息，并且这些信息都是以一种标准格式来呈现。为了实现这个功能，在本任务中将采用 Web Service 的方式来进行开发，将商品查询功能以 Web Service 的形式对外发布。

完成标准：能将商品查询功能以 Web Service 形式进行发布。

应用手段：开发商品查询的 Web Service 服务端。

9.2.2　相关知识

9.2.2.1　Web Service

Web Service 已经作为一种成熟的技术被广泛地应用在互联网中，例如，经常可以在很多网站上看到的天气预报、时刻表以及 IP 地址来源等功能。其实这些网站本身并不能提供天气预报或者提供时刻表，都需要调用第三方提供的专门服务来实现这些应用。而这些第三方提供的专门服务很多都是以 Web Service 的方式来进行发布的。

Web Service 是一种新的 Web 应用程序分支，是自包含、自描述、模块化的应用，具

有发布、定位、通过 Web 调用的特性。Web Service 可以执行从简单的请求到复杂商务处理的任何操作。一旦部署以后，其他 Web Service 应用程序可以发现并调用它部署的服务。Web Service 是一种应用程序，它可以使用标准的互联网协议，如超文本传输协议（HTTP）和 XML，将功能纲领性地体现在互联网和企业内部网上。可将 Web 服务视作 Web 上的组件编程。可以说使用 Web Service 大大缩小了 Web 应用程序之间的相互隔阂，增强了交互性，如图 9-6 所示。

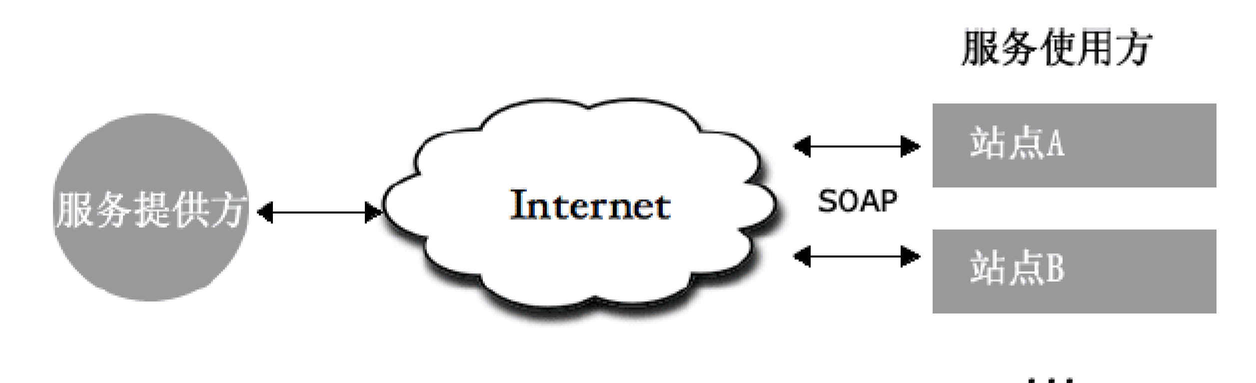

图 9-6　Internet 中的 Web Service

9.2.2.2　Web Service 中的关键技术

Web Service 平台需要一套协议来实现分布式应用程序的创建。任何平台都有它的数据表示方法和类型系统。要实现互操作性，Web Service 平台必须提供一套标准的类型系统，用于不同平台、编程语言和组件模型中的不同类型系统之间进行沟通。目前这些协议有以下 4 条。

1．XML 和 XSD

可扩展的标记语言 XML 是 Web Service 平台中表示数据的基本格式。除了易于建立和易于分析外，XML 主要的优点在于它既与平台无关，又与厂商无关。XML 是由万维网协会（W3C）创建，W3C 制定的 XML SchemaXSD 定义了一套标准的数据类型，并提供了一种语言来扩展这套数据类型。

Web Service 平台是以 XSD 来作为数据类型系统的。当采用某种语言如 VB.NET 或 C# 来构造一个 Web Service 时，为了符合 Web Service 标准，所有使用的数据类型都必须被转换为 XSD 类型。如果还想让它能够在不同平台和不同软件的不同组织间使用，还需要用某种东西将它包装起来，它就是一种协议，如 SOAP 等。

2．SOAP

SOAP 即简单对象访问协议（Simple Object Access Protocol），它是用于交换 XML 编码信息的轻量级协议。它有 3 个主要方面：XML-envelope 为描述信息内容和如何处理内容定义了框架，将程序对象编码成为 XML 对象的规则，执行远程过程调用（RPC）的约定。SOAP 可以运行在任何其他传输协议上。例如，可以使用 SMTP，即因特网电子邮件协议来传递 SOAP 消息。在传输层之间的头是不同的，但 XML 有效负载保持相同。

Web Service 希望在不同的系统之间能够实现用“软件-软件对话”的方式相互调用，打破了软件应用、网站和各种设备之间的格格不入的状态，实现“基于 Web 无缝集成”的功能。

3．WSDL

Web Service 描述语言 WSDL 是用机器阅读的方式提供的一个正式描述文档的 XML 语

言，用于描述 Web Service 及其函数、参数和返回值。因为是基于 XML 的，所以 WSDL 既是机器可阅读的，又是人可阅读的。

4．UDDI

UDDI 是为电子商务建立标准；UDDI 是一套基于 Web 的、分布式的、为 Web Service 提供的、信息注册中心的实现标准规范，同时也包含一组使企业能将自身提供的 Web Service 注册，以使别的企业能够发现的访问协议实现标准。

5．远程过程调用 RPC 与消息传递

Web Service 其实是一种在实现应用程序间的通信方法。现在有两种应用程序通信的方法：RPC 远程过程调用和消息传递。使用 RPC 时，客户端的概念是调用服务器上的远程过程，通常方式为实例化一个远程对象并调用其方法和属性。RPC 系统试图达到一种位置上的透明性：服务器暴露出远程对象的接口，而客户端就好像在本地使用的这些对象的接口一样，这样就隐藏了底层的信息，客户端也就根本不需要知道对象是在哪台机器上。

9.2.2.3　Web Service 带来的好处

实际上，Web Service 采用 9.2.2.2 节中介绍的技术的主要目标是为了实现跨平台的可互操作性。为了达到这一目标，Web Service 完全基于 XML（可扩展标记语言）、XSD（XMLSchema）等独立于平台、独立于软件供应商的标准，是创建可互操作的、分布式应用程序的新平台。由此也可以看到，使用 Web Service 有以下几点优势。

1．跨防火墙的通信

如果应用程序有成千上万的用户，而且分布在世界各地，那么客户端和服务器之间的通信将是一个棘手的问题。因为客户端和服务器之间通常会有一个防火墙或者代理服务器。在这种情况下，不便于把客户端程序发布到每一个用户手中。传统的做法是，选择用浏览器作为客户端，进行 B/S 结构的程序开发，把应用程序的中间层暴露给最终用户，但是这样做开发难度大，程序很难维护。

如果将中间层组件换成 Web Service，就可以从用户界面直接调用中间层组件，从而省去建立 ASP 页面的一步。要调用 Web Service，可以直接使用 MicrosoftSOAPToolkit 或.NET 这样的 SOAP 客户端，也可以使用自己开发的 SOAP 客户端，然后把它和应用程序连接起来。不仅缩短了开发周期，还减少了代码复杂度，并能够增强应用程序的可维护性，同时，应用程序也不再需要在每次调用中间层组件时，都跳转到相应的“结果页”。

2．应用程序集成

企业级的应用程序开发者都知道，企业里经常都要把用不同语言写成的、在不同平台上运行的各种程序集成起来，而这种集成将花费很大的开发力量。应用程序经常需要从运行在 IBM 主机上的程序中获取数据；或者把数据发送到主机或 UNIX 应用程序中去。即使在同一个平台上，不同软件厂商生产的各种软件也常常需要集成起来。通过 Web Service，应用程序可以用标准的方法发布功能和数据，供其他应用程序使用。

3．B2B 集成

用 Web Service 集成应用程序，可以使公司内部的商务处理更加自动化。但当交易跨越

供应商和客户、突破公司的界限时会怎么样呢？跨公司的商务交易集成通常叫做B2B集成。Web Service 是 B2B 集成成功的关键。例如，把电子下单系统和电子发票系统发布出来，客户就可以以电子的方式发送订单，供应商则可以以电子的方式发送原料采购发票。当然，这并不是一个新的概念，EDI（电子文档交换）早就是这样了。但是，Web Service 的实现要比 EDI 简单得多，而且 Web Service 运行在 Internet 上，在世界任何地方都可轻易实现，其运行成本就相对较低。不过，Web Service 并不像 EDI 那样，是文档交换或 B2B 集成的完整解决方案，它只是 B2B 集成的一个关键部分，还需要许多其他的部分才能实现集成。

用 Web Service 来实现 B2B 集成的最大好处在于可以轻易实现互操作性。只要把商务逻辑发布出来，成为 Web Service，就可以让任何指定的合作伙伴调用这些商务逻辑，不管他们的系统在什么平台上运行、使用什么开发语言。这样就大大减少了花在 B2B 集成上的时间和成本，让许多原本无法承受 EDI 的中小企业也能实现 B2B 集成。

知识点小贴士

当然 Web Service 也有它不适合的场合，例如，在开发桌面程序或者局域网内的应用程序时，使用 Web Service 就不是一个很好的主意，不过这些场合也不是 Web Service 在给出时所设定的应用目标。

9.2.2.4 .NET 的 Web Service 解决方案

微软的.NET 技术是时下最为流行的 Web Service 开发技术。首先，因为其公司的产品占有相当大的市场份额，以致新推出的.NET 得以有比较稳定的用户群；其次，也是更重要的是.NET 平台不仅延续了微软一贯的编程风格，而且还增加了许多支持 Web 服务的关键性技术，使得.NET 在操作的简单性和执行的稳定性、高效性上结合得非常好。

在 Visual Studio 2005 开发工具中，可以在网站项目中直接添加 Web Service，如图 9-7 所示，然后在其基础上按照规范进行开发，其开发非常便捷。同时，在 Visual Studio 2005 中开发基于 Web Service 的应用也十分便捷。

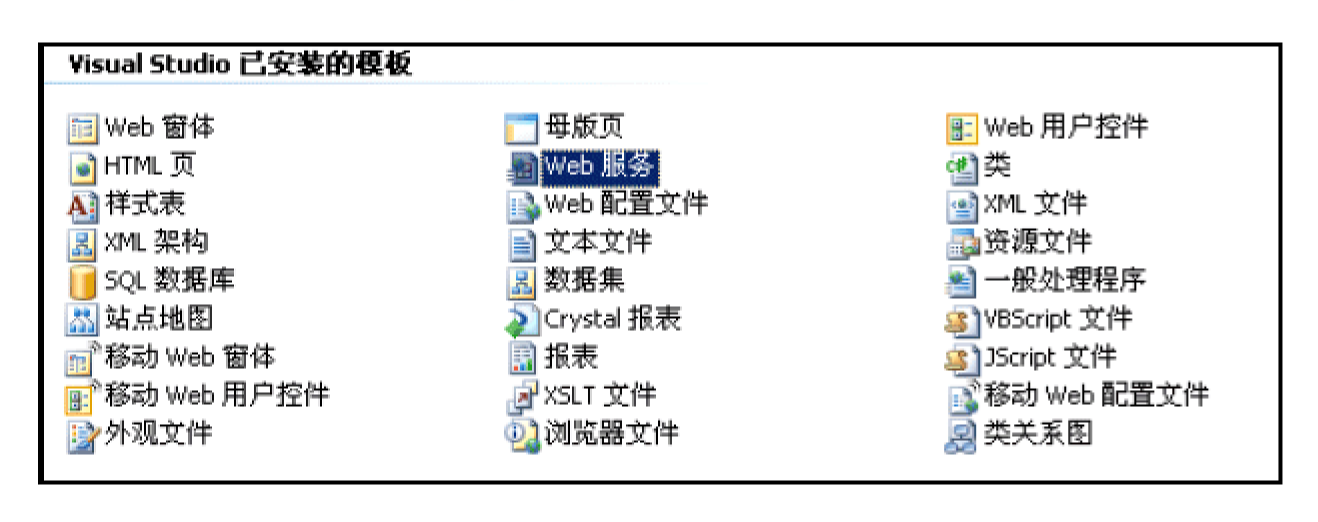

图 9-7 添加 Web Service 新项

9.2.3 任务实施

1．添加 Web Service 新项

在现有的网站项目中添加新项，选择“Web 服务”选项，然后指定 Web Service 的名

称即可，这里指定 Web Service 新项的名称为“ItemInfo.asmx”，如图 9-8 所示。

图 9-8　指定用于商品信息查询的 Web Service 项的名称

单击“添加”按钮后会在网站项目中增加两个文件 ItemInfo.asmx 以及 ItemInfo.cs，分别如图 9-9 和图 9-10 所示。其中，.asmx 文件是 Web Service 的服务文件，用户可以通过对该文件的访问来实现对 Web Service 的访问；而自动生成到 App_Code 文件夹中的.cs 文件则是 Web Service 的实现类以及方法。

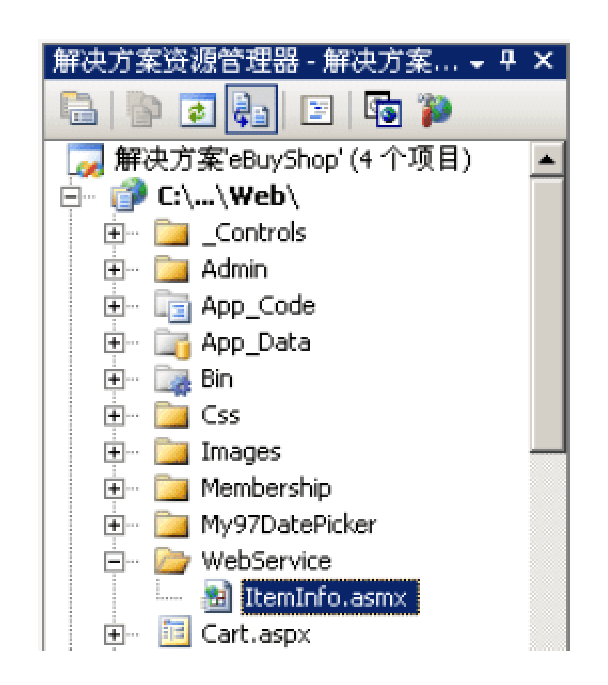

图 9-9　添加的 ItemInfo.asmx 文件

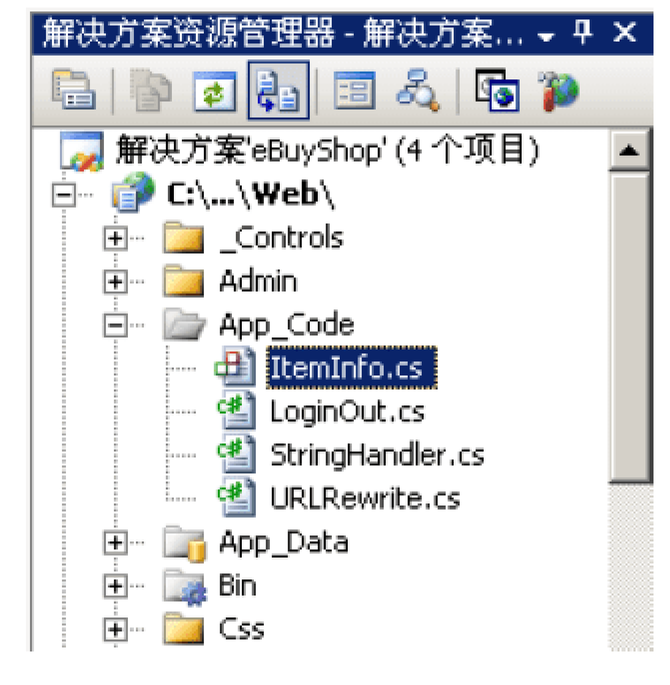

图 9-10　添加的 ItemInfo.cs 源文件

在.cs 的源文件中生成的默认代码如下：

```
using System;
using System.Web;
using System.Collections;
using System.Web.Services;
using System.Web.Services.Protocols;

/// <summary>
/// ItemInfo 的摘要说明
/// </summary>
[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
public class ItemInfo : System.Web.Services.WebService
```

```
{
    public ItemInfo()
    {
        //如果使用设计的组件，请取消注释以下几行代码
        //InitializeComponent();
    }
    [WebMethod]
    public string HelloWorld()
    {
        return "Hello World";
    }
}
```

其中，[WebMethod]是 Web Service 自己提供的特性，它表示其下的方法是该 Web Service 提供的 Web 方法，是允许 Web 使用 SOAP 协议对它们进行调用的。一个 Web Service 文件中可以有多个[WebMethod]，一个[WebMethod]只能对它下面的一个方法有效，在自动生成的代码中就已经包含了一个名为“HelloWorld”的 Web 方法。

2．实现商品信息查询的 Web 方法

清楚了 Web Service 的大致原理后，就可以开始实现商品信息查询的 Web Service 了。这里，主要需要在 Web Service 中开发一个 Web 方法，该方法能够通过接收来自调用方的关键字从商品信息库中查找到相应的商品，并以 XML 的形式返回查询结果。为此，需要编写如下的 Web 方法来实现商品的查询：

```
[WebMethod]
public List<Item> getItem(string keyword)
{
    IList<Item> iItems = ItemManager.SearchItems(keyword);        //调用逻辑层的查询方法
    List<Item> items = new List<Item>();
    foreach (Item item in iItems)
    {
        items.Add(item);
    }
    return items;                                                  //将查询的结果以列表的方式返回
}
```

Web 方法编写好后，如何来检测其是否工作正常呢？可以直接运行网站项目，或者将网站项目在 Web 服务器上部署后进行访问。在访问 ItemInfo.asmx 时，可以在浏览器中看到如图 9-11 所示的页面。

图 9-11 所示的页面样式是由.NET 自动生成的，显示在最上面的“ItemInfo”表示该 Web Service 的名称，同时在页面可以看到有个 getItem 链接，这个链接事实上就是前面定义的 Web 方法的名称，单击这个链接即可直接调用 getItem 方法。单击此链接后，可以看到如图 9-12 所示的页面。

图 9-11　访问 ItemInfo.asmx 的页面

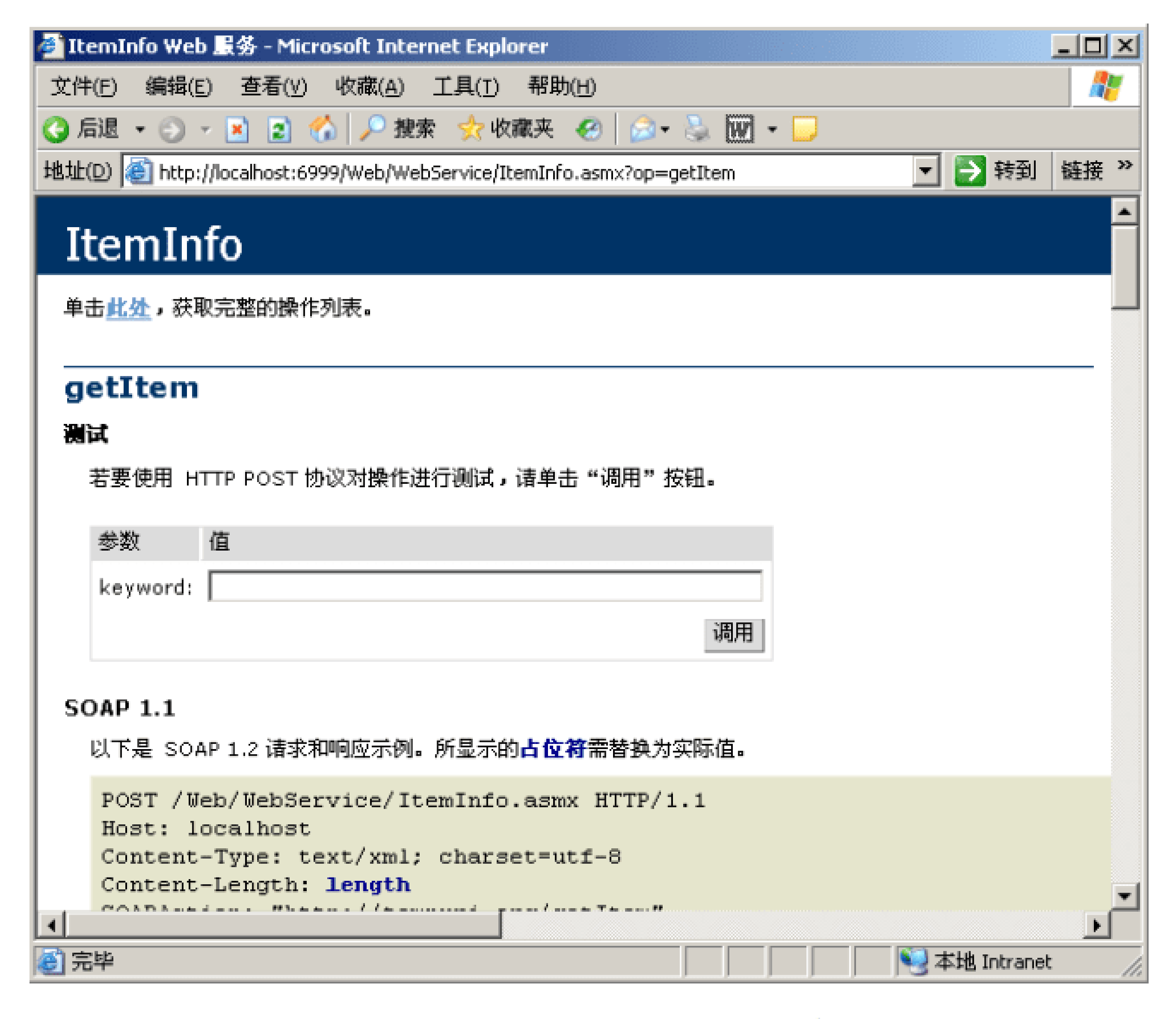

图 9-12　调用 getItem 方法

在图 9-12 中可以看到其中有一个文本框以及一个"调用"按钮，这里的参数"keyword"其实就是在定义 Web 方法时给定的方法传入参数的名称，在这里只需在文本框中输入对应的值，然后单击"调用"按钮就可以将参数的值传递给 Web Service 中的 getItem 方法并完成调用。例如，在文本框中输入"s"后，调用之后的结果如图 9-13 所示。在调用后浏览器会新开一个窗口来显示调用后的结果，此时浏览器地址栏中内容也有所变化。

可以看到 Web 方法的返回结果是以 XML 的方式进行呈现的，这也是 Web Service 为什么能够跨平台进行集成的原因。由于在 getItem 方法中查询结构以列表的方式返回，因此在返回结果的 XML 中可以看到有多组<Item></Item>标签，其中的每一组<Item></Item>都表示一个匹配查询关键字"s"的查询结果，因为这些商品的名称中都包含有字母"s"。

此时，商品信息查询的 Web Service 发布完毕，可以被用来进行调用了。

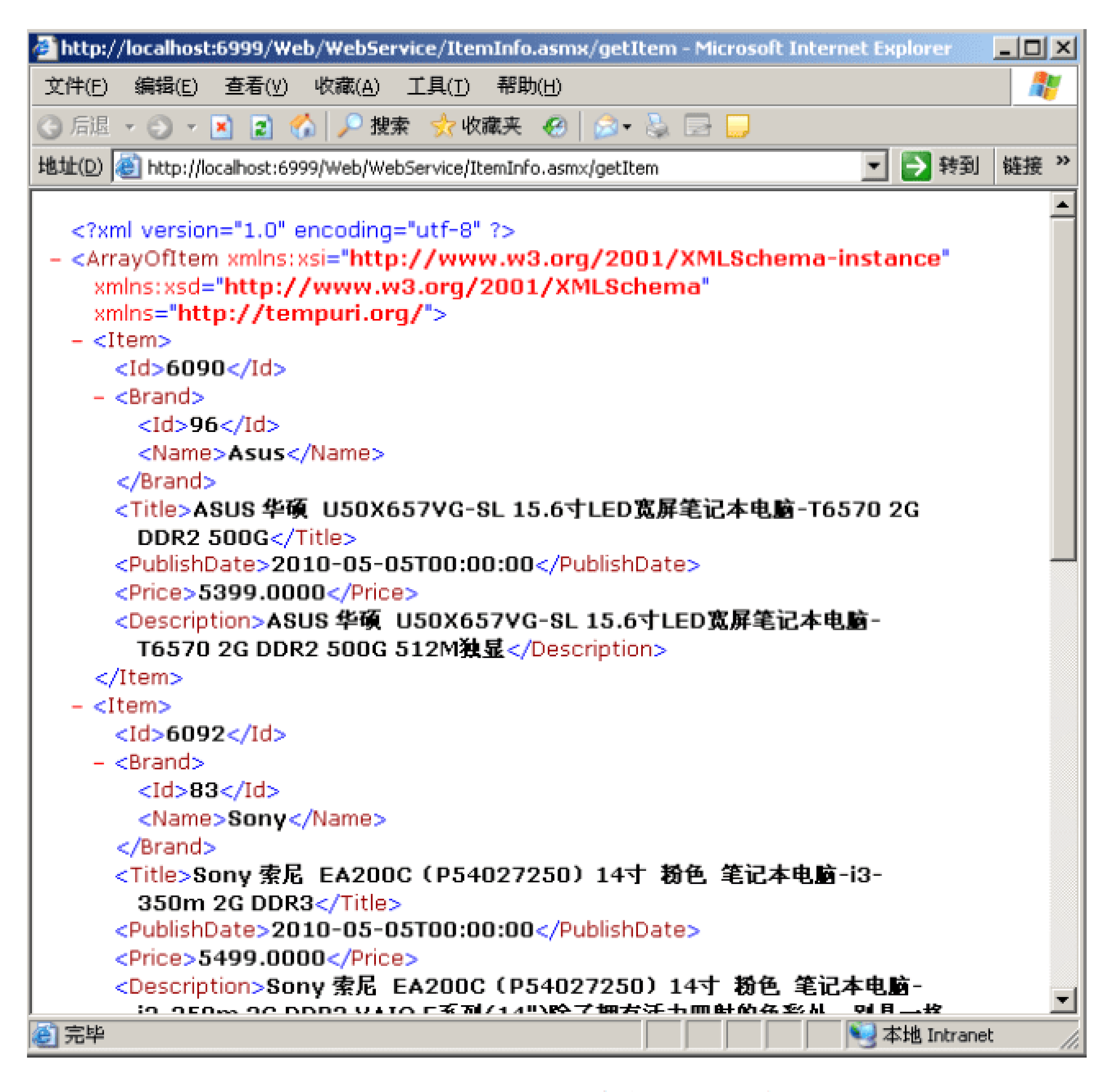

图 9-13　Web 方法的调用结果

在图 9-11 所示的页面中，还有一个非常重要的链接是“服务说明”，单击此链接后可以查看 Web Service 的一些（例如输入/输出）参数的定义，打开后可以看到如图 9-14 所示的页面。服务说明同样以 XML 的方式进行呈现，在地址栏中可以看到 WSDL 的字样，事实上就是采用 WSDL（Web Services Description Language）的方式来对 Web Service 进行描述的，这种方式的描述能被其他的程序在集成时识别。其中，主要对 Web Service 中的 Web 方法，以及方法的输入输出参数的个数类型进行了描述。

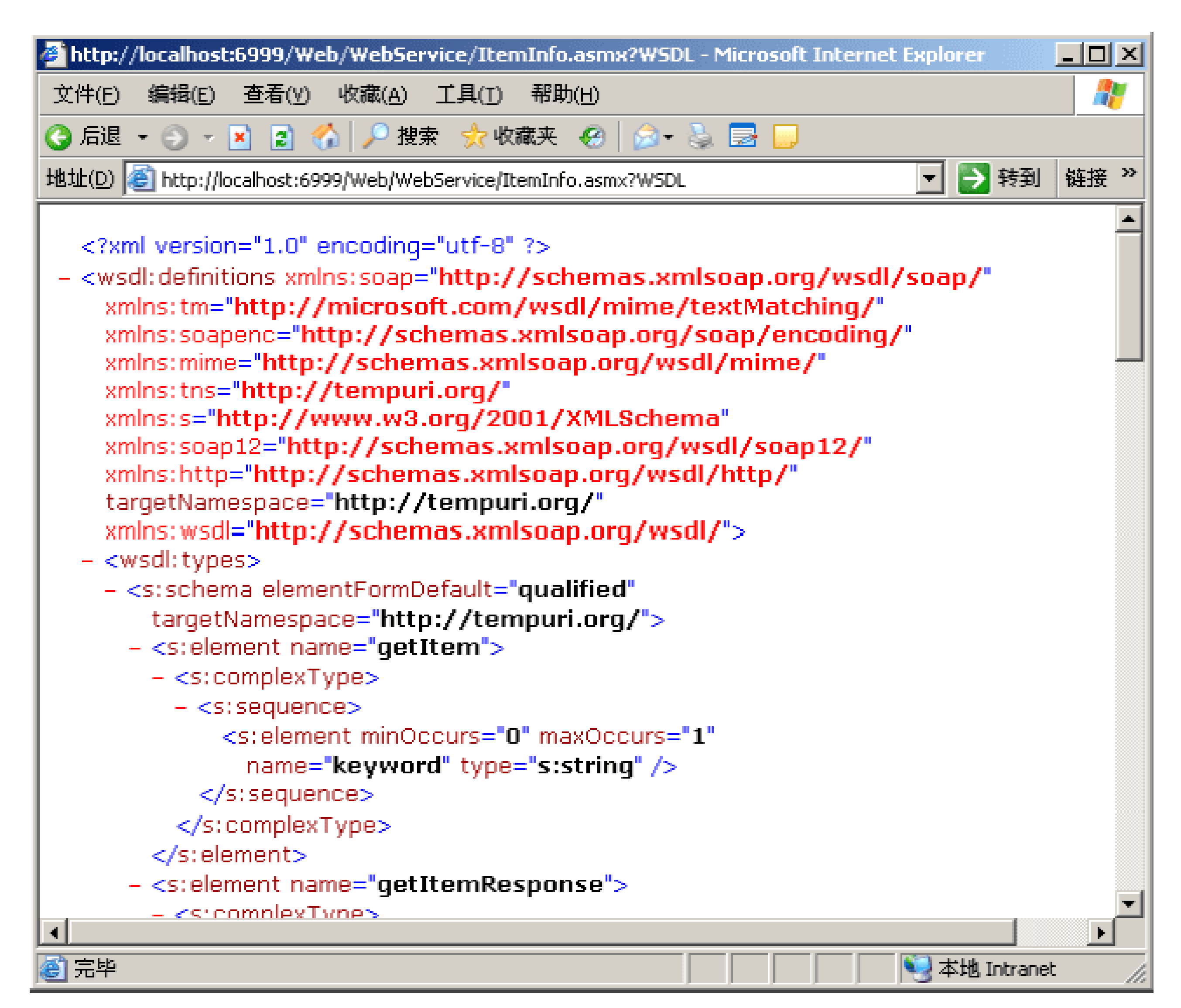

图 9-14　服务说明

3．调用 Web Service

Web Service 开发并发布好后如何被使用被调用呢？以下是模拟调用的过程。下面将新建一个网站项目来模拟需要使用该 Web Service 的用户，然后看看如何在他们的网站中集成该 Web Service 的功能。

（1）右击网站项目，在弹出的快捷菜单中选择“添加 Web 引用”命令，如图 9-15 所示。

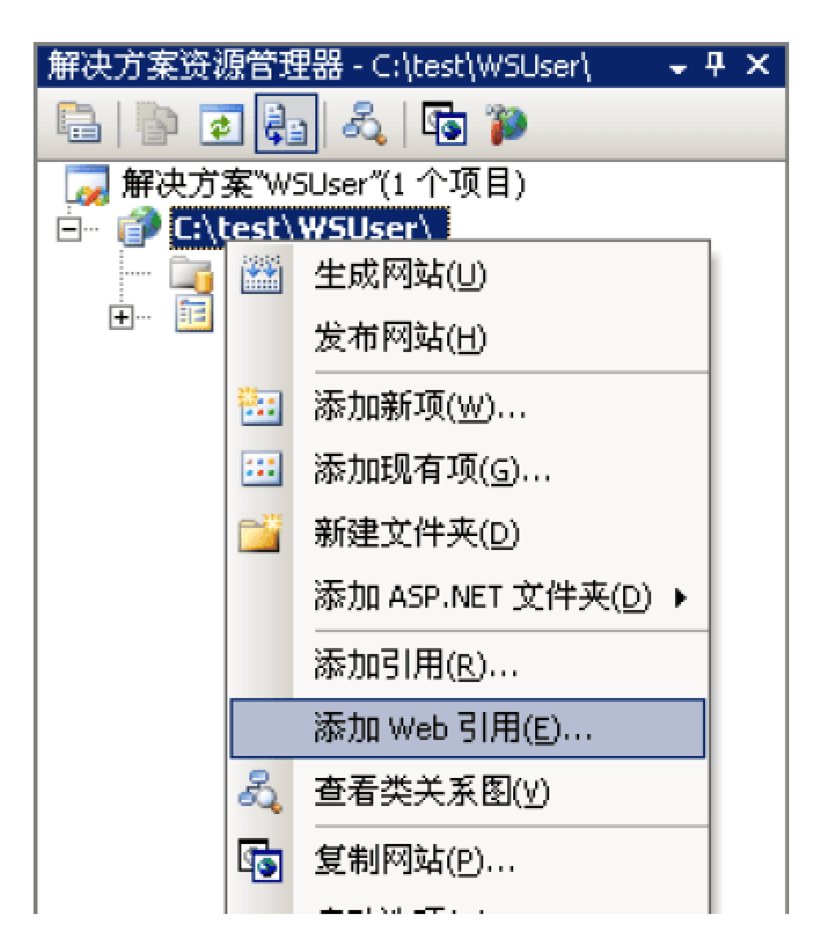

图 9-15　选择“添加 Web 引用”命令

（2）将需要引用的 Web Service 的网址复制到打开的对话框的 URL 文本框中，如图 9-16 所示。

图 9-16　复制 Web Service 网址

（3）单击“前往”按钮，可以看到先前开发好的名为 ItemInfo 的 Web Service，同时也可以看到在其中可以被引用的 Web 方法，将对话框中的“Web 引用名”项目中的内容由“localhost”改为“WSItem”，如图 9-17 所示。

图 9-17　修改 Web 引用名

知识点小贴士

“Web 引用名”是调试 Web Service 方法时需要使用的，它可以看作是在客户端产生的一个 Web Service 的类，通过它就可以按需要对 Web Service 进行访问。

（4）单击“添加引用”按钮后就完成了对 Web Service 的引用，同时可以看到在资源管理器中生成了 App_WebReferences 文件夹以及增加的相关文件，如图 9-18 所示。

（5）Web Service 引用后就可以使用 Web Service 中的方法来实现需要的功能了。在 Default.aspx 中设计如图 9-19 所示的样式，该页面通过调用易购商城提供的商品查询 Web Service 来查询商品信息。

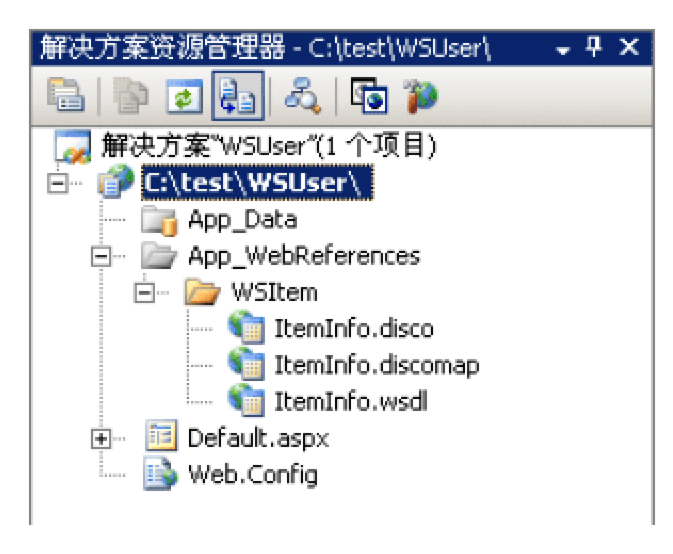

图 9-18　引用 Web Service 后生成的文件

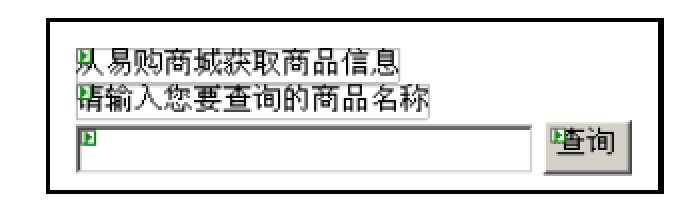

图 9-19　Web Service 调用端的页面样式

（6）对“查询”按钮编写如下的代码，使其能够实现对易购商城商品查询 Web Service 的调用。

```
protected void Button1_Click(object sender, EventArgs e)
{
    String keyword = tbKeyWord.Text;
    WSItem.ItemInfo service = new WSItem.ItemInfo();
    WSItem.Item[] items = service.getItem(keyword);
    for (int i = 0; i < items.Length;i++ )
    {
```

```
            Response.Write(items[i].Id +"<br>");
            Response.Write(items[i].Title +"<br>");
            Response.Write(items[i].Price +"<br>");
            Response.Write(items[i].PublishDate +"<br>");
        }
}
```

（7）在文本框中输入“s”，单击“查询”按钮，即可看到如图 9-20 所示的页面效果。

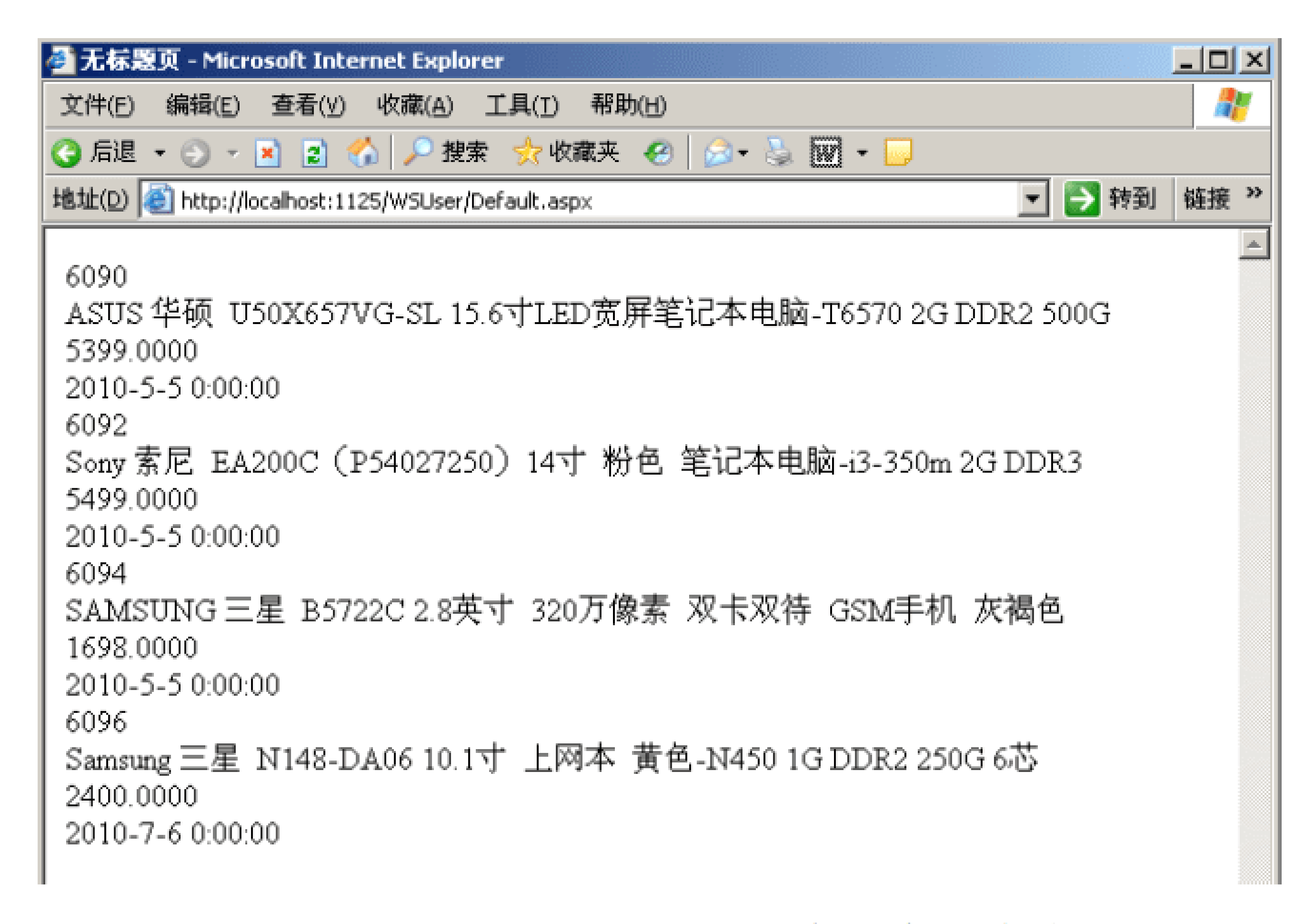

图 9-20　调用 Web Service 实现商品查询

通过这种方式，实现了通过调用 Web Service 来进行商品信息查询的功能，实际上查询功能本身是由易购商城提供的，在模拟的引用端则可以采用调用的方式将该功能集成到引用端的网站中去。

9.2.4　任务评价

标　　准	掌 握 情 况
能理解 Web Service 的工作原理	熟练□　基本□　模糊□
能进行 Web Service 服务端的开发	熟练□　基本□　模糊□
能进行 Web Service 引用端的开发	熟练□　基本□　模糊□
自 我 小 结	

9.2.5　任务拓展

读者可以在互联网上根据 IP 地址查找所在地区的 Web Service，并将该 Web Service 加

入到易购商城的网站中，根据用户的 IP 地址对用户的所在地进行判断。

本章小结

本章主要阐述了两种用于 Web 开发的高级技术：HttpHandler 以及 Web Service，以及分别用这两种技术实现了图片的动态数字水印和查询功能的对外发布。通过 HttpHandler 的学习，读者可以了解.NET 对于 HTTP 请求处理的流程和机制，并掌握进行 HttpHandler 开发的一般方法。通过对 Web Service 的学习，读者也了解了 Web Service 的原理和工作机制，并能掌握使用.NET 进行 Web Service 开发的方法以及在.NET 中引用外部已有的 Web Service 来实现特定功能的方法。

自我检测

一、选择题

1．对于 HttpHandler 程序的说法，不正确的是（ ）。

A．HttpHandler 程序的后缀名是.ashx

B．HttpHandler 是 HTTP 请求处理的终点

C．HttpHandler 必须实现 IHttpHandler 接口

D．如果 IsReusable 属性设置为 False 时，HttpHandler 程序只能使用一次

2．下列关于 HttpModule 以及 HttpHandler 的说法，正确的是（ ）。

A．HttpModule 可以有多个，但是 HttpHandler 只能有一个

B．HttpModule 不执行任何操作，HttpHandler 才是真正的执行者

C．HttpModule 可以做判断，HttpHandler 用于执行请求

D．请求到达 HttpHandler 之前，可能会被某个 HttpMoudle 抛弃

3．Web Service 文件的扩展名是（ ）。

A．.aspx　　B．.ascx

C．.asmx　　D．.ashx

4．下列关于 Web Service 的描述，不正确的是（ ）。

A．Web Service 可以穿越防火墙进行通信

B．Web Service 的返回结果是 XML 格式

C．我们只能调用自己开发的 Web Service，而不能调用其他网站的

D．Web Service 的描述语言 WSDL 是 XML 格式

5．Web Service 的通信协议是（ ）。

A．DCOM　　B．SOAP

C．CORBA　　D．UDP

6．The following statements on Web Service, which is wrong?（ ）

A．We can only use the Web Service which developed by us, but can not use the Web Service which provided by other website

B．The format of the result of Web Service is XML

C．The format of WSDL is XML

D．Web Service can communicate with other through fireworks

二、简答题

1．简述使用.NET 进行 Web Service 开发的一般步骤。

2．简述 HttpHandler 的工作原理，以及开发 HttpHandler 应用的一般步骤。

第 10 章　系统配置和部署

技能目标

1．能使用 web.config 文件进行站点配置。
2．能使用站点管理工具对站点进行配置。
3．能使用 VS 发布工具进行站点发布。

相关词汇

英 文 单 词	中 文 含 义	英 文 单 词	中 文 含 义
debug		configuration	
deny		globalization	
authentication		deliver	
deploy		duration	

工作任务

任务 10.1　配置数据库连接
任务 10.2　配置身份验证和授权
任务 10.3　使用发布工具进行站点部署

任务 10.1　配置数据库连接

10.1.1　任务分析

任务目标：当前系统中，数据库的连接是通过一个数据库连接字符串来实现的，它被直接写在数据访问层的一个数据库访问类中。但在实际的使用中，这种做法有很大的局限性：当数据库连接发生改变或者采用不同的数据库时，需要对数据库访问类重新进行编译和部署，在使用上不是很灵活。由于 web.config 文件可以用来保存参数，因此可以考虑把数据库连接字符串保存在 web.config 文件中。本任务的目标就是采用这种方式来改进数据库连接字符串的保存和读取方式。

完成标准：能够从 web.config 文件中读取数据库连接字符串以实现数据库连接，当连接对象发生改变时不需要重新编译站点程序。同时，要求对保存在 web.config 中的连接字符串进行加密。

应用手段：使用 web.config 中的<connectionStrings>标记来实现。

10.1.2　相关知识

配置文件是可以根据需要进行更改的 XML 文件。开发人员可以根据开发和运行的需要来更改和使用配置文件而不必重新编译程序。ASP.NET 提供两种配置文件：machine.config 和 web.config，它们都是基于 XML 格式的配置文件，只是其配置的作用域不同。

machine.config 用于将计算机范围的策略应用到本地计算机上运行的所有.NET Framework 应用程序。默认的 machine.config 配置文件存储在计算机的 C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\CONFIG 目录下，也被称为服务器配置文件。

web.config 用来存储 ASP.NET Web 应用程序的配置信息（如身份验证），一般出现在 ASP.NET 应用程序的根目录中。其实，在所有的文件夹中都可以有一个 web.config 文件。当新建一个 Web 应用程序后，默认情况下会在根目录中自动创建一个 web.config 文件，包括默认的配置，所有的子目录都继承它的配置。如果需要修改子目录的配置，可以在子目录下新建一个 web.config 文件，它可以提供除从上级目录继承的配置信息以外的配置信息，也可以重写或修改上级目录中定义的设置。因此，在 Web 服务器上存在一个由 machine.config 和多个 web.config 配置文件组成的配置文件系统，如图 10-1 所示。

在运行时对 web.config 文件的修改不需要重新启动服务器即可生效。web.config 文件可以扩展，通过自定义配置参数并编写配置标记处理程序可对它们进行处理。

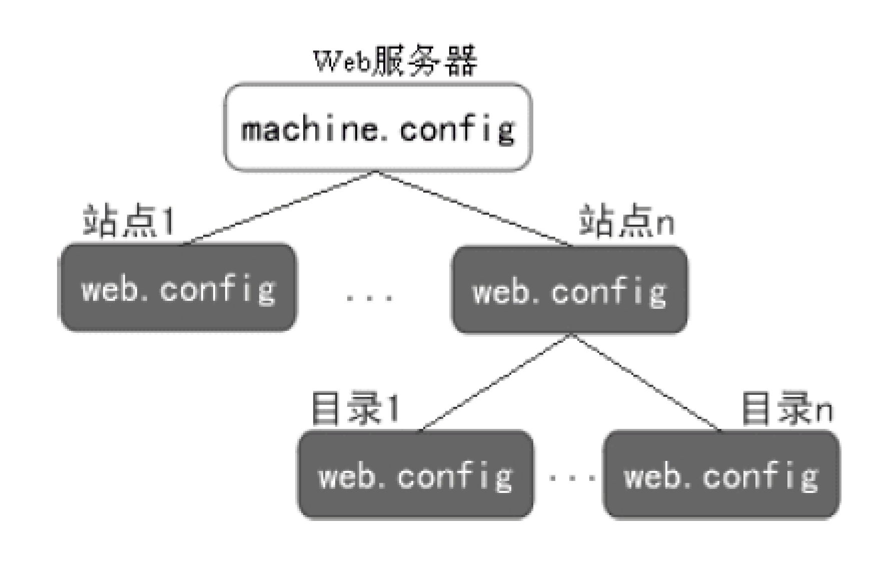

图 10-1　配置文件层次结构

web.config 配置文件中包含有如下的一些基本标记。

1．<appSetting>标记

此标记用于存储自定义的应用程序的配置信息，如文件路径、XML Web Service URL 或存储在应用程序的.ini 文件中的任何信息，例如：

```
<appSettings>
    <add key="AdminLogin" value="/Admin/Adminlogin.aspx" />
</appSettings>
```

2．<compilation>标记

此标记用于指定编辑应用程序源文件所采用的编译选项。例如，可以指定编辑期间所需的程序集的引用，它还可以指定是否在运行应用程序，例如：

```
<compilation debug="true" />
```

3．<customErrors>标记

当应用程序发生错误时，通过该标记可以将客户端的浏览器重新定向到指定的错误页，而不是直接向用户显示一个错误信息，例如：

```
<customErrors defaultRedirect="~/ErrorPage.htm"    mode="RemoteOnly">
      <error statusCode="404" redirect="FileNotFound.htm" />
</customErrors>
```

4．<authentication>标记

此标记用于验证访问者的身份，只有拥有合法的用户名和密码才能进入。其具体的配置将在任务 10.2 中进行阐述，例如：

```
<authentication mode="Forms">
      <forms name="Login" loginUrl="~/Admin/AdminLogin.aspx" />
</authentication>
```

当然还有其他的很多用于各种不同用途的标记，这里不一一指出。在该任务中，将采用<connectionStrings>标记来实现数据库连接字符串的存储。

10.1.3　任务实施

1．在 web.config 中配置数据库连接字符串

数据库连接字符串属于站点级的配置，可以将其存储在 web.config 文件中。与站点配置相关的信息都保存在标记<configuration>和标记</configuration>之间，可以在其中添加一组<connectionStrings>标记来保存一个连接字符串，其具体的内容如下：

```
<connectionStrings>
      <add  name="eBuyShop"  connectionString="Data  Source=localhost;Initial  Catalog  =eBuyShop;User
ID=sa;password=123456" />
</connectionStrings>
```

通过这段代码就可以配置一个数据库连接字符串。其实还可以在其中添加多个<add>标记，用于存储多个不同的数据库连接配置。<add>标记主要的属性如下：

- name：用于标记连接字符串的名称。
- connectionStrings：用于存储连接字符串。

2．在程序中访问连接字符串

以前是直接将数据库连接字符串编写在 DBHelper 类中，现在就需要修改程序，让 DBHelper 类能够读取保存在 web.config 中的连接字符串。可通过如下代码来实现连接字符串的读取：

```
String connectionString=
ConfigurationManager.ConnectionStrings["eBuyShop"].ConnectionString;
```

这里的["eBuyShop"]就是在<add>标记中指明的连接字符串的名称；

需要注意的是，要使 DBHelper 能够访问 web.config 中的信息，需要在数据访问层项目中添加 System.Configuration 类的引用，并在 DBHelper 类中加入相应的命名空间的引用：

```
using System.Configuration;
```

这样，在以后数据库连接发生变化时，就不需要对源代码进行修改和重编译，只需修改 web.config 中的相关配置即可。

3．加密和解密数据库连接字符串

web.config 文件在程序运行时是不需要编译的，因此保存在其中的信息都是以明文形式存在的。由于数据库连接字符串中存储了数据库的用户名和密码等重要信息，为了保证数据库的安全，需要对连接字符串进行加密，使其明文不具可读性。

ASP.NET 自身提供了一个命令工具 aspnet_regiis.exe，可以用来实现连接字符串的加密和解密。其加密的语法如下：

```
aspnet_regiis.exe -pef section physical_directory -prov provider
```

其中，各个参数的含义如下：

- section：表示要加密的标记部分。
- physical_directory：用于指定站点的物理路径。
- provider：指定加密提供的程序。

按照这个语法，对系统中的配置文件进行加密的命令就是：

```
aspnet_regiis.exe -pef "connectionStrings" "项目的路径" -prov "DataProtection ConfiguratonProvider"
```

其中，DataProtection ConfiguratonProvider 是 Windows 提供的数据保护程序，它使用 Windows 内置的密码技术来实现加密。在命令行工具中运行上述命令后即可完成连接字符串的加密，加密后的代码如下：

```
<connectionStrings configProtectionProvider="DataProtectionConfigurationProvider">
<EncryptedData>
<CipherData>
<CipherValue>AQAAANCMnd8BFdERjHoAwE/Cl+sBAAAAmfIXhiY4oEivSzs8m9sLeQQAAAACAA
AAAADZgAAqAAAABAAAAAGTugIQZ2uG88OYRhMR+vrAAAAAASAAACgAAAAEAAAACp
KHmNCc88HulMDyhlzgkDYAQAAU76n1OAaEnts/TxQqAj90EtQLx6P9MtnEBp/Z+3F0aFHIZAbnNS4
0FQ5RzmAgA9jovOdvYVunmTcJw+K9W69P+KG0W/9PGtXMCMZEkg6KmO2RB4akgq5OX4HJGaK
HZ3tmgSbL5Xkkt5zumvy7MUXSc+bfeiWaRYL5OTZEbGoLWElJSIecOSe62N9+LUuJFNd7y79k4UXu
ODAlmec1IEeb6pYNd9HoHUbvNMwiNGaYLZrG36gCh5suY4BEAZCSoqr7MHgbBAECY8WJDYNoir
9ygkJiemG4g/s5NHRuOycdfpKOcr3JCdSZA6BaXPlucJC86REfkv6DBD3CBveICi8STm9qA4IhRch2V2o
n1jvQ3TjMzogfYkGMDvc3hjmw7T1VRDQbTMMecWBPncQpTQRr9/p3Uq8yyO8tJXYIrGDywaW+6X
AGh7o5qgMOsTeP8FHtcywQrRQnQ5Qyk/5pEwbnWS+EIXWiCONMmcm0Si7m7042Og5bv/kBrJqJIfv/
AKkWp22YWib5XraOIDNhlsJV7mFMEb1JesLyVDNmf7ONhO4W8URiHpPOx57FTvv+feYN8oBdFQ
MVsBRZDahi67kNY2ZGPshAdJTE6Kns/1x94Fp2KSeG6j5gRQAAADWBW6ZyPyZag2hDU3u7ahPZvg
ECw==</CipherValue>
</CipherData>
</EncryptedData>
</connectionStrings>
```

知识点小贴士

加密的语法还有另外的一种形式：

```
aspnet_regiis.exe -pe section -app virtual_directory -prov provider
```

其中，virtual_directory 表示当前应用程序的虚拟路径。

对于加密后的连接字符串，也可采用如下的命令来进行解密：

```
aspnet_regiis.exe -pdf "connectionStrings" "项目的路径"
```

进行解密时，不需要指定相应的解密提供程序就可以正常解密。解密后，连接字符串将恢复到加密前的样子。

问题思考

既然知道了加密和解密的命令，如果碰到别人加密过的 web.config 配置文件，可以直接采用解密命令进行解密吗？

10.1.4　任务评价

标　　准	掌 握 情 况
能在配置文件中配置连接字符串	熟练□　基本□　模糊□
能在代码中获取配置文件中的连接字符串	熟练□　基本□　模糊□
能对连接字符串进行加密和解密	熟练□　基本□　模糊□
自 我 小 结	

10.1.5　任务拓展

如果系统需要在使用 SQLServer 数据库的同时也能够使用 Access 数据库，读者可修改 web.config 配置文件以及源代码，使得程序能够在两种不同的数据库之间进行切换。

任务 10.2　配置身份验证和授权

10.2.1　任务分析

任务目标：在开发系统登录页时，采用了 Session 对象来保存用户的登录信息，通过判断 Session 中保存的登录信息判断用户是否已经完成登录。但这种做法也有一个弊端，需要

在系统的每个页面上都加上对 Session 进行判断的代码来验证用户是否登录。为了解决这个麻烦，ASP.NET 提供了一套基于用户和角色的安全体系。本任务的目标就是通过对 ASP.NET 的安全体系进行配置来实现非编码的用户验证。

完成标准：通过安全配置来实现用户登录的验证。

应用手段：使用 VS 提供的站点管理工具对站点进行访问配置。

10.2.2　相关知识

10.2.2.1　身份验证

ASP.NET 提供的安全体系主要由验证和授权两个部分组成，它们分别对应于 web.config 配置文件中的标记<authentication>以及标记<authorization>。

身份验证是确认用户身份的过程，通常通过凭证或者一些身份验证表单来实现。例如，可以使用 Session 来完成身份验证。在进行验证配置时，可以采用 4 种不同的验证方式，如表 10-1 所示。

表 10-1　身份验证的方式

验 证 方 式	描　　述
Windows	将 Windows 验证指定为身份验证方式，这个是默认值
Forms	设计一个验证页，所有的验证请求都将被重定向到这个页面
Passport	将 Microsoft Passport 身份验证作为身份验证，是一种集中式商业验证模式
None	不指定任何身份验证，允许匿名访问或自行编码实现验证

Windows 身份验证提供程序用于验证登录，并将它们映射到 Windows 组。这种验证方式在实际应用中不多见。

Forms 验证是最常见的一种验证方式。在内部，它使用 Cookie 来跨页维护状态，当用户提供有效的用户名和密码通过验证后，其身份验证 Cookie 将写入到该用户的计算机上，应用程序中的每一个页面在被访问时都首先对 Cookie 进行检查，通过判断 Cookie 的值来判定用户是否已经登录。

Forms 标记具备 5 个属性和一个子标记，如表 10-2 所示。

表 10-2　Forms 标记

属　性　值	描　　述
Name	指定用于身份验证的 Cookie 的名称，默认为.ASPXAUTH
loginUrl	指定登录页的 URL，如果没有找到任何有效的身份验证 Cookie，客户端将被重新定向到该页，默认值为 default.aspx
protection	指定 Cookie 使用的加密类型，有 All、None、Encryption 以及 Validation 4 种
Timeout	指定以整数分钟为单位的时间，超过该时间 Cookie 将失效，默认为 30 分钟
Path	指定 Cookie 的路径，默认为“/”，表示该 Cookie 可用于整个站点
<credentials>	在配置文件中用于指定要访问站点的用户名和密码，并包含用来指定加密格式的 passwordFormat 属性

另外一种安全机制是授权，用于控制对 URL 资源的客户端访问，可以在任何级别（计算机、站点、应用程序、子目录或页）上声明。在本书中的项目中，管理员后台的所有页面都存储在站点根目录的 Admin 子目录中，可以在该文件夹的配置文件中对访问权限进行配置。通过<authorization>标记可以进行授权配置，另外，还可以通过子标记<allow>和<deny>进行允许和拒绝授权。

10.2.2.2 用户授权

用户授权是指确定经身份验证的用户是否可以访问可以通过请求资源的过程。在 ASP.NET 2.0 中，主要包括如下授权选项。

1．文件授权

文件授权是通过 FileAuthorizationModule 类来实现的。这种方式类似于 UNIX 下的文件权限检查，但更加严格和完善。当用户请求某个页面时，该类负责对执行.aspx 或.asmx 处理程序文件的访问控制列表进行检查，以确定用户是否拥有权限访问受保护的文件。这种授权方式需要通过系统管理员对文件权限的设置来实现。

2．URL 授权

URL 授权是通过 UrlAuthorizationModule 类来实现的。即使用该类可以有选择地允许或者拒绝访问用户、角色和谓词（get、pest 等 Http 谓词）对 URL 命名空间任意部分的访问。这种授权方式对于用户的页面请求，并不是从文件权限出发，而是根据系统的配置情况，决定用户的请求是否是经过授权的。采用 web.config 文件中关于授权和角色的配置节来实现。

3．基于角色的授权

基于角色的授权是从逻辑上实现用户与授权的分离，在代码中可以调用显式的角色检查方法，如 User.IsInRole 以及 Roles.IsUserInRole 等，或者使用 PrincipalPermission 请求以确保特定角色的成员。

知识点小贴士

在实际使用中，URL 授权和基于角色的授权应用最为广泛，很多时候两者一起使用。例如，首先为用户设计角色，然后再为角色进行 URL 授权，这样可大大提高授权管理的效率和可控性。

10.2.3 任务实施

1．使用站点管理工具配置验证类型

Web 站点管理工具可对 Web 应用程序的多个方面进行管理和设置，是一个基于 Web 的应用程序。具体操作步骤如下：

（1）通过 IDE 的“网站”菜单下的“ASP.NET 配置”命令，或通过“解决方案资源管理器”的工具栏来打开站点管理工具，如图 10-2 和图 10-3 所示。

图 10-2　IDE 中“网站”菜单

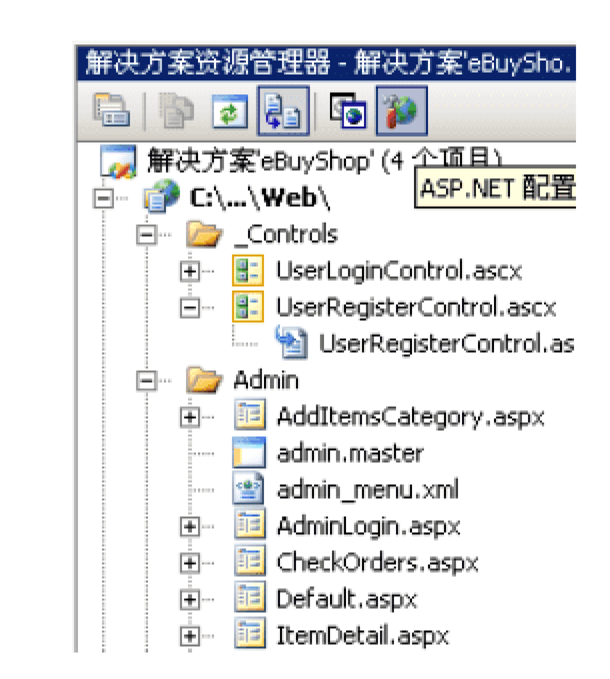

图 10-3　解决方案资源管理器的工具栏

（2）站点管理器打开后，可以看到如图 10-4 所示的欢迎界面。

图 10-4　站点管理器的欢迎界面

（3）此界面提供了 4 个选项卡，即“主页”、“安全”、“应用程序”和“提供程序”。在此将使用“安全”选项卡中的功能来完成身份验证类型的选择。选择“安全”选项卡，如图 10-5 所示。

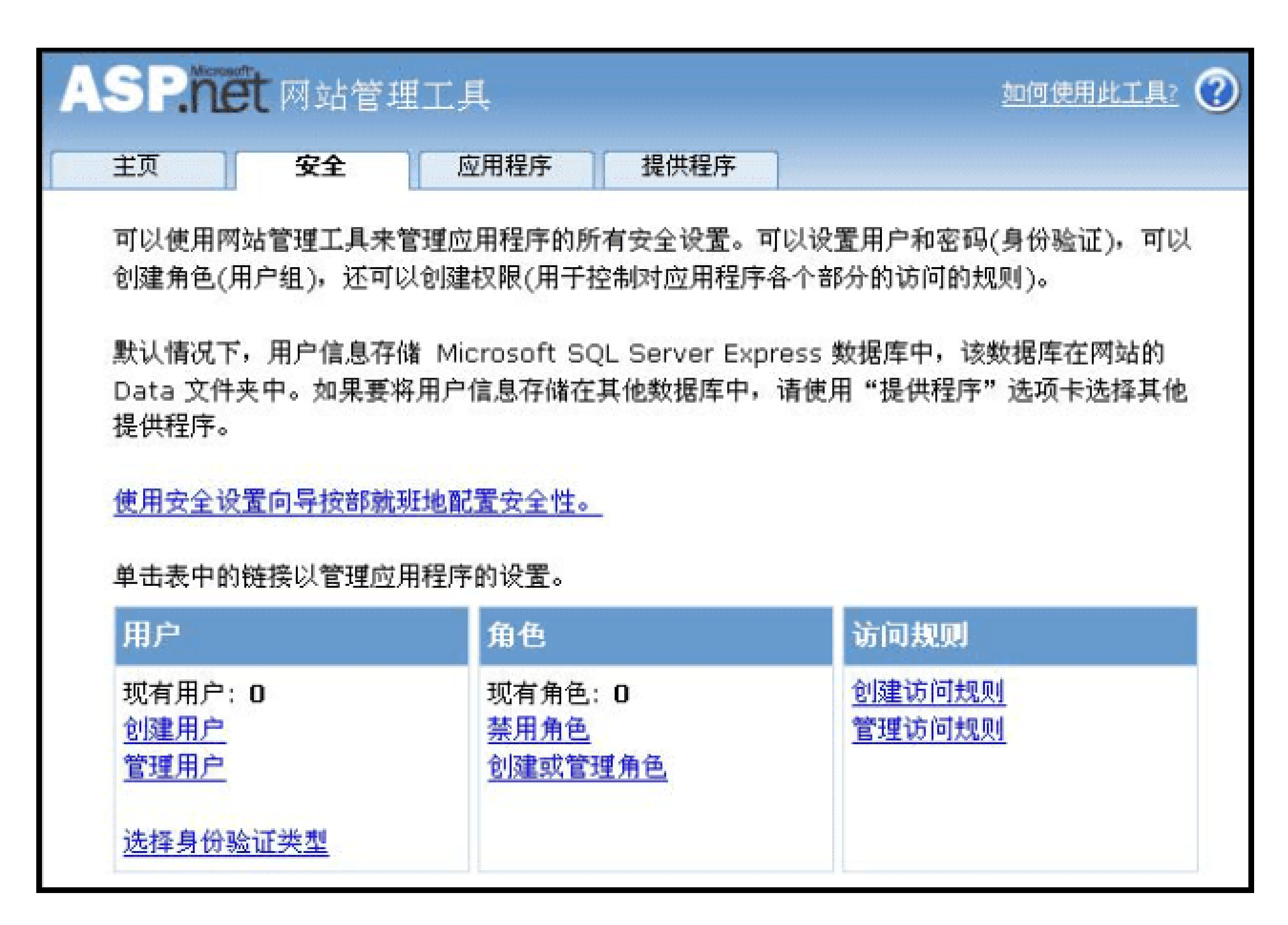

图 10-5　选择“安全”选项卡

（4）在“用户”栏中单击“选择身份验证类型”链接，进入验证类型选择界面，如图 10-6 所示。

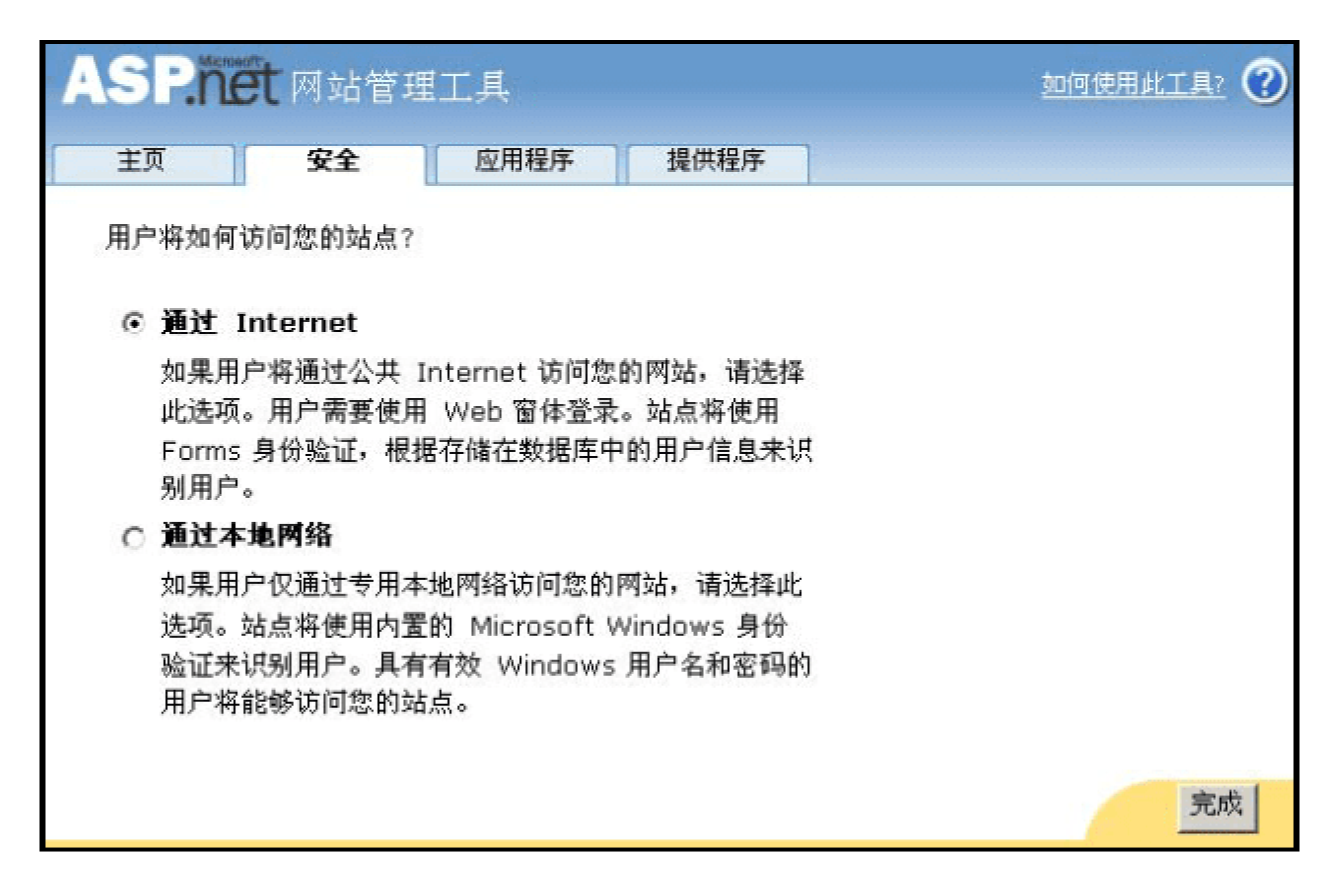

图 10-6　选择验证类型

（5）选中“通过 Internet”单选按钮，表示用户是通过公共的互联网来访问站点，用户需要使用 Web 窗体登录，也就是采用 Forms 身份验证。单击“完成”按钮完成选择。这时，可以在 web.config 文件里面看到如下的代码：

```
<authentication mode="Forms">
    <forms name="Login" loginUrl="~/Admin/AdminLogin.aspx" timeout="60" />
</authentication>
```

知识点小贴士

“应用程序”选项卡中提供了许多与应用程序相关的配置，包括应用程序设置、SMTP 邮件服务器设置、调试和跟踪设置、整个 Web 应用程序的启动和调试以及默认页定义等。

“提供程序”选项卡中可配置网站管理数据的存储方式，并可用数据库的方式进行保存。

2．使用站点管理工具控制 Admin 目录的访问

Admin 文件夹中保存的页面是提供给管理员进行系统管理的。因此，需要对这个文件夹的访问进行控制，以确保只有管理员才能访问这个目录下的页面。在站点管理器的“安全”选项卡中，选择“访问规则”栏目中的“创建访问规则”选项，可以打开如图 10-7 所示的页面。在页面的左侧是当前站点下的目录结构，右侧是可供选择的访问规则。

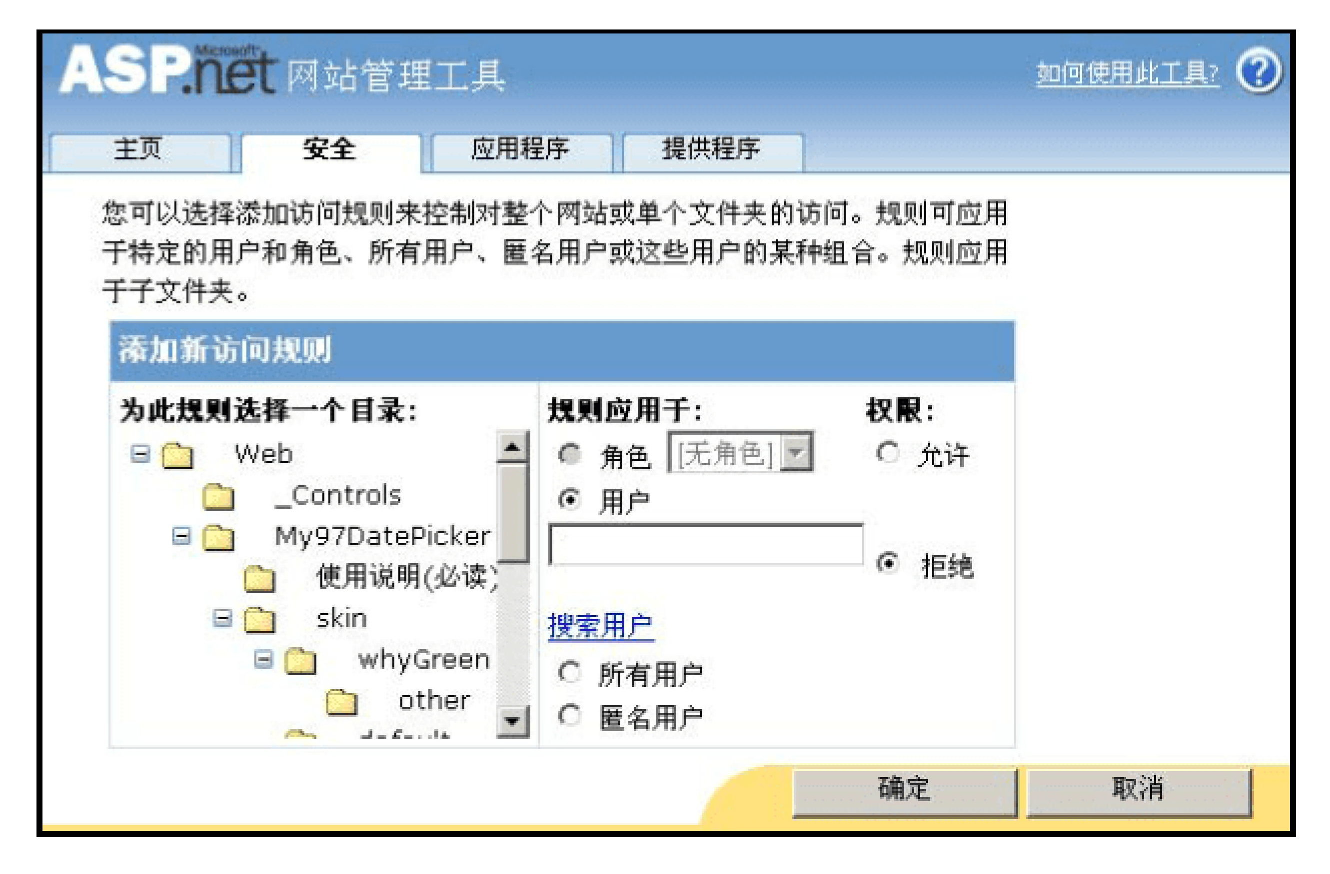

图 10-7　创建对目录的访问规则

在左边的目录中选择 Admin，右边的“规则应用于”选择“用户”单选项并在下面文本框中输入 admin，“权限”设置为允许，通过这种方法就可以将用户 admin 设定为允许访问 Admin 目录，而其他的用户则不可以访问。另外，还可以在如图 10-5 所示的“安全”选项卡中管理访问规则。

10.2.4　任务评价

标　　准	掌 握 情 况
能实现基于 Forms 形式的身份验证	熟练□　基本□　模糊□
能对目录 Admin 进行访问控制	熟练□　基本□　模糊□
自 我 小 结	

10.2.5　任务拓展

参照本节所讲的方法，在站点中添加一类角色 manager，这类角色可以对 Admin 目录进行访问；同时，把若干个用户归为 manager 角色的成员，使它们能够访问 Admin 目录中的内容。

任务 10.3　使用发布工具进行站点部署

10.3.1　任务分析

任务目标：系统在开发完成后需要以某种形式进行交付，所谓“交付”，就是将系统交给用户，由用户自己来运行系统。Web 应用程序完成后，也需要进行交付，使其能够在非开发环境下运行起来。这时，就需要对它进行部署，将它部署在指定的 Web 服务器使其能够稳定地运行。本任务的目标就是采用 Visual Studio 2005 自带的发布工具对 Web 站点进行部署。

完成标准：将 Web 应用程序部署在 Web 服务器上，使其能够顺利地运行。

应用手段：使用 Visual Studio 2005 的发布工具来完成。

10.3.2　相关知识

读者可能都比较熟悉“安装程序”这种说法。一般而言，软件或应用程序都是采用安

装的方式使其能够在用户的计算机上运行。在这里所说的部署是和安装有一定的区别：安装是指将应用程序包装成易于部署的形式，通过一个可执行文件方便地进入目标计算机；部署则是获得应用程序并将它放置到另一台计算机上的过程。

知识点小贴士

普通桌面程序的安装通常是采用一个可执行程序将程序文件复制到指定的文件夹中。然后通过修改操作系统的一些参数，使其能够运行。这些用于安装的可执行程序一般由专门的安装程序开发工具来完成，如 InstallShield 等。

10.3.3 任务实施

1．部署前的准备

在进行部署之前，需要做如下一些准备工作。

（1）在 web.config 中关闭调试功能。如果不关闭调试功能，它将把一些调试信息插入编译好的 DLL 中，从而降低应用程序的性能。

（2）选择 Release 的方式编译应用程序，如图 10-8 所示。

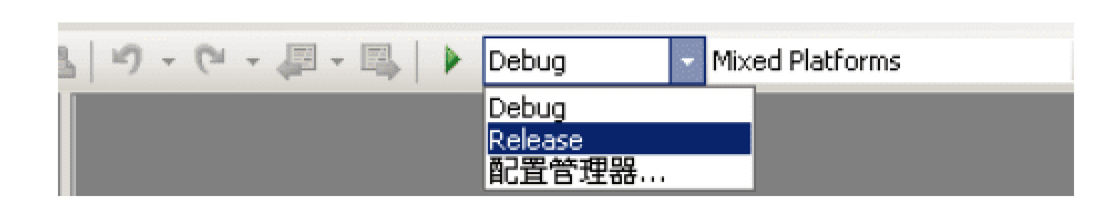

图 10-8　选择 Release 编译模式

2．部署站点

对于 Web 应用程序而言，只需要将页面和程序复制到 Web 服务器的指定目录下就可以让程序运行起来，这里可以通过几种不同的方式来进行部署。

（1）复制站点。复制站点的方式最为简单，直接将程序所在的文件夹复制到 Web 服务器的指定目录下即可。通常的做法是，先在 Web 服务器上建好虚拟目录，然后进行站点文件的复制。采用这种方式会将所有的页面文件、资源文件、源代码等内容复制到站点下。

（2）使用 Visual Studio 发布工具来进行部署。单击“解决方案资源管理器”工具栏上的“复制网站”按钮，打开 Visual Studio 自带的发布工具，如图 10-9 所示。

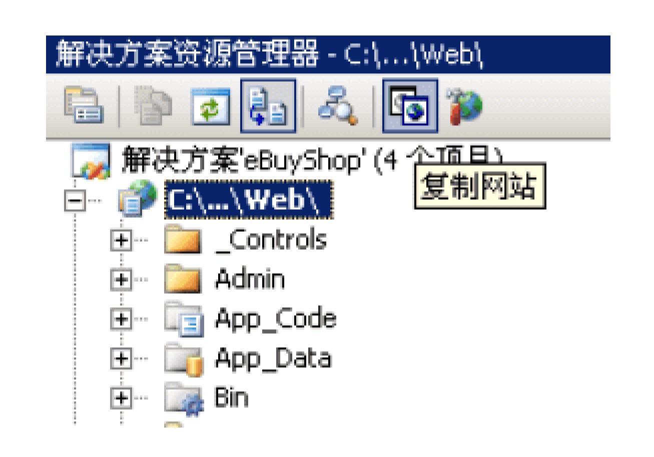

图 10-9　单击“复制网站”按钮

复制网站工具同样可以将文件复制到另外一个指定的位置。其中，有 4 个发布位置可供用户选择，包括文件系统、本地 IIS、FTP 站点和远程站点等，如图 10-10 所示。

图 10-10　复制站点

单击“连接”按钮，即可打开发布位置选择的对话框，如图 10-11 所示。

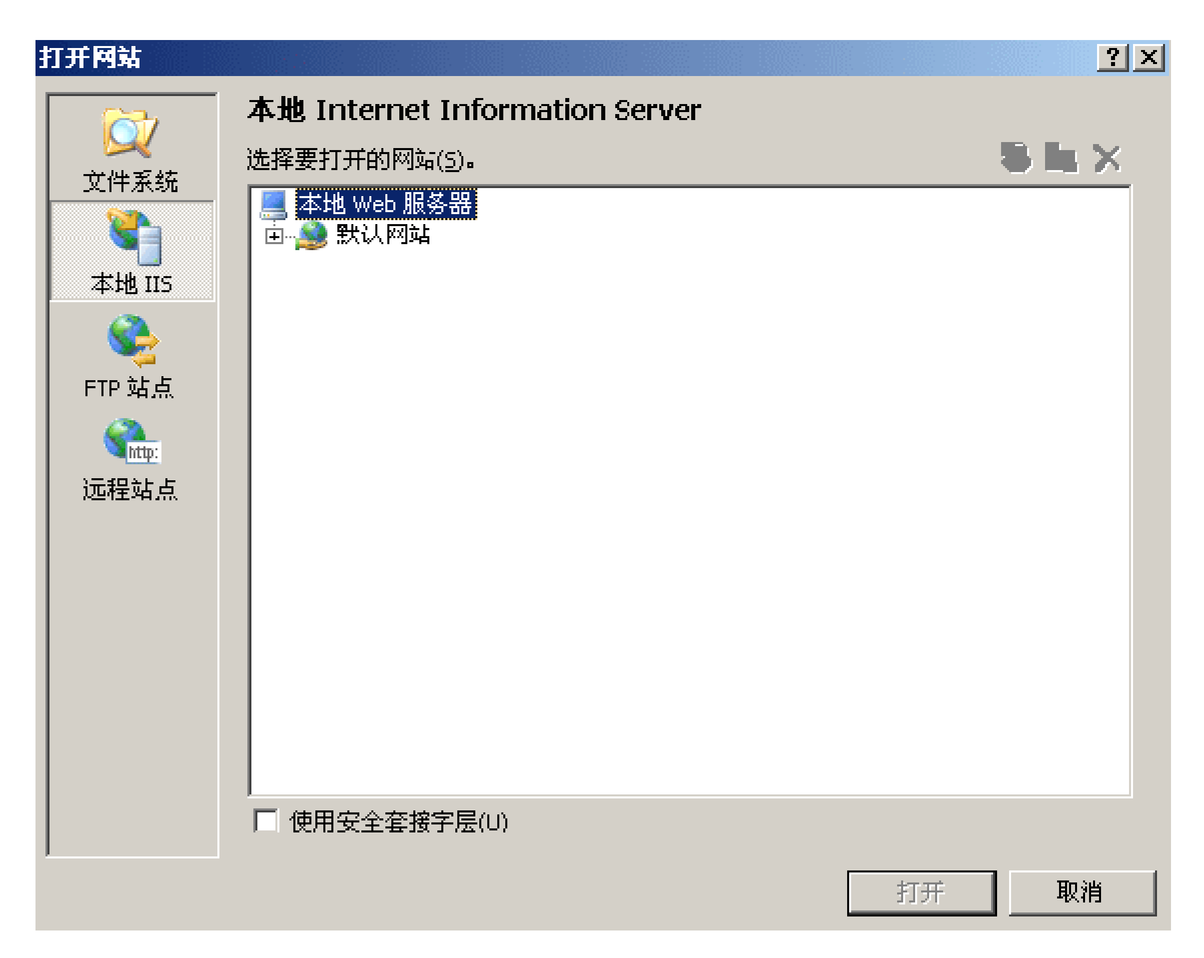

图 10-11　选择发布位置

复制站点时，可以在图 10-11 中左边栏目中选择要发布的文件，将它们放置到右边的栏目中，这些文件就会被部署在指定的位置。同时 IDE 还可以自动检查是否有文件更新，也可以通过查看日志来查看以前的部署记录。

知识点小贴士

“文件系统”方式进行部署其实就是将应用程序复制到指定的文件夹。

本地 IIS：若 Web 服务器采用本机，则可以采用本地 IIS 方式将应用程序部署到本地的 IIS 服务器上。

FTP 站点：通过 FTP 的方式将应用程序部署到特定的服务器上，虚拟主机一般都采用 FTP 的方式来上传程序。

远程站点：直接通过 HTTP 的方式将应用程序部署到远程服务器上。

（3）预编译部署。预编译部署是最常见的站点部署形式，也是一种比较好的部署形式。ASP.NET 的页面在第一次被访问时反应速度通常比较慢，因为它有一个编译的过程。为了

缩短这个过程，可以采用预编译的方式来进行部署。在使用预编译部署方式时，每个页面已经被编译为 DLL 文件和一些占位符文件，因此在部署时不必再包含.cs 的源文件。右击“解决方案资源管理器”中的 Web 项目，在弹出的快捷菜单中选择“发布网站”命令，即可发布预编译站点，如图 10-12 所示。

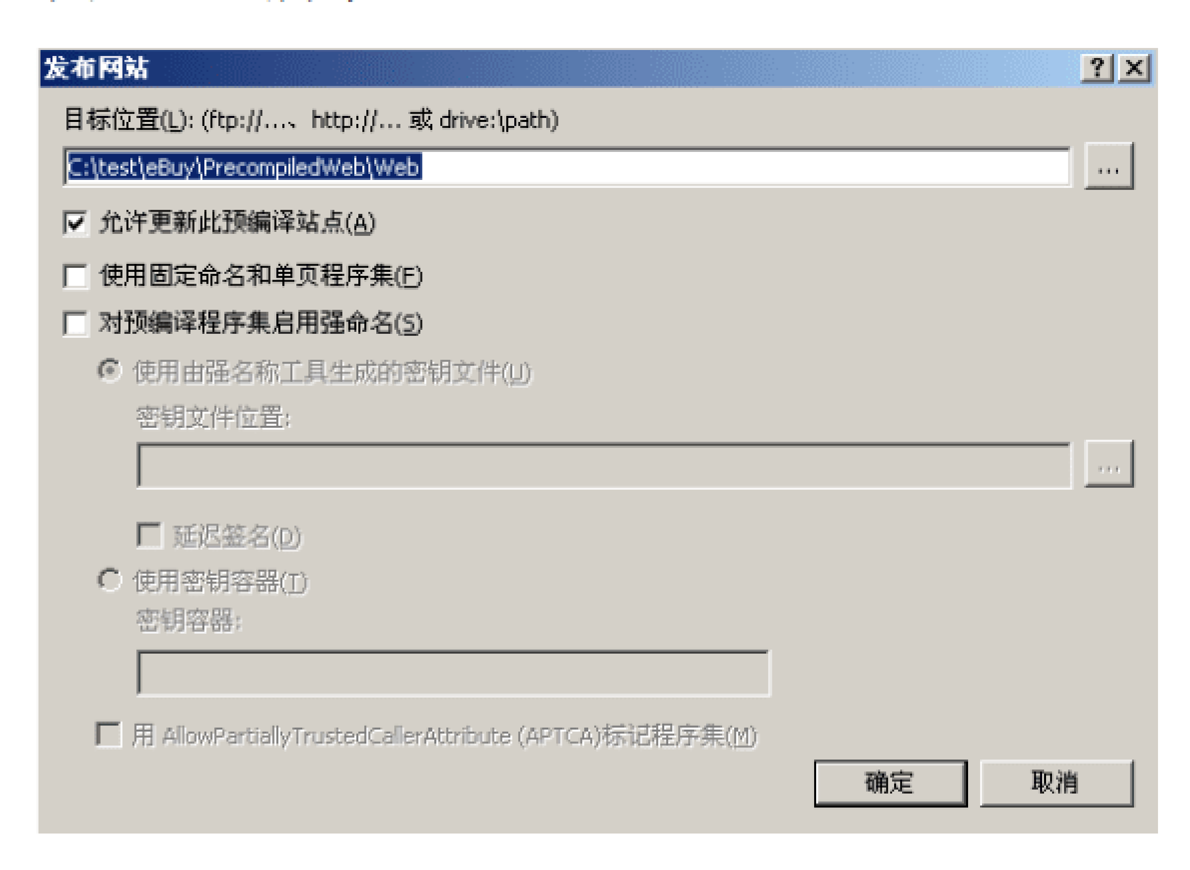

图 10-12　发布预编译站点

在图 10-12 中，3 个复选框的作用分别如下：

- 允许更新此预编译站点：选中该复选框，表示将编译和复制站点，但不对.aspx 页面进行任何修改；如果取消选中该复选框，则页面中的所有代码会被剥离，并放置到 DLL 文件中。如果页面发生了修改，则需要重新编译和部署才能在站点中进行更新。
- 使用固定命名和单页程序集：取消选中该复选框，表示编译过程将所有的页面代码和后台代码打乱编译成多个 DLL 文件，代码的保密性提高。但如果页面和代码以后需要修改，则建议选中此项。
- 对预编译程序集启动强命名：选中该复选框，表示可以指定生成 DLL 过程中使用的键，预编译过程创建的 DLL 文件就会用选中的键来签名。

10.3.4　任务评价

标　　准	掌 握 情 况
能通过复制的方式部署站点	熟练□　基本□　模糊□
能使用 XCopy 部署站点	熟练□　基本□　模糊□
能使用 Visual Studio 发布预编译站点	熟练□　基本□　模糊□
自 我 小 结	

10.3.5　任务拓展

读者可以采用 3 种不同的方式来完成对站点的部署，通过比较这 3 种方式的使用方法以及后期的维护方法，总结它们各自的特点。

本章小结

作为整个项目完成前的最后一步，配置和部署是项目从开发阶段走向使用阶段的一个关键步骤。本章主要阐述了使用配置文件来保存数据库连接字符串以及访问的方法，并且讲解了对连接字符串进行加密和解密的方法。同时本章也讲解了使用站点管理工具来实现站点的身份验证、授权的基本设置方法以及采用不同的对站点进行部署的方法。在配置和部署时开发者要充分考虑整个项目的运行环境、实际使用要求以及日后可能的维护情况，这些前提都会对配置和部署的方式有影响。

自我检测

一、选择题

1．很多情况下，需要将被保护的页面分类并放置在不同的目录下，这样做是为了（　）。

A．划分功能　　B．便于管理

C．便于编辑　　D．便于调用

2．在 web.config 配置文件中对于子目录 Manage 有如下的一段代码：

```
<authorization>
    <deny users= "?"/>
</authorization>
```

其作用是（　）。

A．只有管理员可以访问 Manage 目录

B．所有匿名用户都可以访问 Manage 目录

C．所有匿名用户都不能访问 Manage 目录

D．所有的用户都不能访问 Manage 目录

3．下列关于授权的说法中，不正确的是（　）。

A．使用授权有利于提高管理效率

B．Forms 验证是授权的一种方式

C．通过文件授权，可以控制对某个目录的访问

D．授权可通过 web.config 中的<authorization>配置节来实现

4．如果需要部署站点，可采用的方式是（　）。

A．使用 Web 站点管理工具　　B．使用文件复制

C．使用内置发布工具　　D．使用 XCopy 命令

5．下列关于部署的说法中，正确的是（　）。

A．部署前需要对站点打包，并提供安装程序

B．部署时可以直接将站点文件复制到服务器上

C．发布预编译站点可以将源代码编译成 DLL 文件

D．直接复制和发布预编译站点的效果一样

6．The following statements on authorization，which is wrong?（　）

A．Authorization can be used to improve the efficiency of identity management

B．We can control the access to the directories through file authorization

C．We can implement authorization through the <authorization> node in web.config file

D．There are three types of authorization in ASP.NET 2.0

二、简答题

1．简述直接复制部署和发布预编译站点部署之间的区别。

2．简述身份验证的几种方式，以及它们之间的区别。

3．简述授权的几种方式，以及它们之间的区别。

第 11 章　系 统 测 试

技能目标

1. 能使用测试工具对系统进行 Web 测试。
2. 能使用测试工具对系统进行负载测试。

相关词汇

英 文 单 词	中 文 含 义	英 文 单 词	中 文 含 义
duration		load testing	
validation		response time	
verification		cache	
performance		stress	

工作任务

任务 11.1　对系统进行 Web 测试
任务 11.2　对系统进行负载测试

任务 11.1　对系统进行 Web 测试

11.1.1　任务分析

任务目标：系统测试是保证系统质量和可靠性的关键步骤，是对系统开发过程中的系统分析、设计和实现的最后复查。在本任务中，将使用 VSTS（Visual Studio Team System）的测试工具对已经完成的易购商城系统进行 Web 测试，检验其 Web 应用程序的功能是否满足各项性能指标要求。

完成标准：对系统的登录模块进行 Web 测试，观察并记录测试结果。

应用手段：使用 VSTS 的测试工具进行 Web 测试。

11.1.2　相关知识

11.1.2.1　系统测试

系统测试的目的是希望能以最少的人力和时间发现系统中潜在的各种错误和缺陷。系

统测试是根据开发过程中各阶段的需求、设计等文档或程序的内部结构精心设计测试实例，并利用这些实例来运行程序，以便发现错误的过程。测试是系统开发过程中一个独立且非常重要的阶段，如何组织好测试，特别是如何选择测试用例，对保障系统质量、降低测试费用有着重要意义。一个高效的测试，有时成为“高产”测试，即用所设计的少量测试用例，发现被测试程序中尽可能多的问题。

在系统测试过程中，需要设计一批测试的输入数据、运行流程以及预期输出结果，这就是所谓的设计测试用例。利用这些测试用例去运行程序，从中发现系统中还存在的问题。基于测试目的，对广义的系统测试而言，它应该遵循以下的几项原则。

（1）加强对测试重要性的认识，要把测试作为系统开发和实施当中一项非常重要的步骤来完成。

（2）确定输入数据和预期输出结果是测试用例必不可少的部分。特别是后者，只有将实际输出结果与预期输出结果进行比较，才知道程序是否有问题。在 Web 测试中，这种预期结果不仅仅是输出的值，它还有可能是跳转到的页面等情况。

（3）应避免开发人员自己检查自己的系统。有些错误是开发人员自己对问题理解上有偏差，这样的错误，开发人员自己很难检查出来。但这并不意味着开发人员对自己所开发的系统就不做任何测试，一般在每编写完一段程序代码时就要进行测试，这就是所谓的单元测试。

（4）对测试出来的结果进行仔细检查，以免遗漏错误信息。

（5）在设计测试用例时，除了要设计一些合法的输入条件外，也应设计一些非法的输入条件，这样可以发现程序在执行异常输入时的反应。

（6）在测试的各个阶段要做好测试计划，并严格按照计划执行。

（7）测试用例、测试计划、测试统计和测试报告都应保留，以便日后发现错误并实施改进。

11.1.2.2 测试的方法

系统测试的方法主要有以下 5 种。

（1）黑箱测试。即不管系统内部是如何实现的，只从系统外部根据设计要求对系统以及模块进行测试。

（2）数据测试。即用大量实际测试数据进行测试。数据的类型要齐全，除了正常的值以外还要覆盖各种“边值”和“断点”。

（3）穷举测试。即对系统运行的各个分支都进行测试。

（4）操作测试。即检查从操作到各种显示、输出是否与设计要求相一致。

（5）模型测试。即核对所有的计算结果。

在测试的不同阶段由于测试对象的不同，所采用的测试方法也不一样。

11.1.2.3 测试的主要步骤

系统测试的步骤主要如下。

（1）单元测试。即根据详细设计的说明，测试模块的重要控制路径，力求在模块范围

内发现错误。在这个阶段一般采用白盒测试方法。

（2）集成测试。即根据总体设计中各功能模块的说明以及制定的测试计划，将经过单元测试的模块逐步进行组装和测试，把每个已经测试的模块并入到系统整体中。在这个阶段一般采用黑盒测试方法。

（3）确认测试。即按照需求分析说明书中定义的全部功能和性能要求以及确认测试计划，来测试整个系统是否达到了要求。在这个阶段一般也采用黑盒测试方法。

知识点小贴士

测试的方法主要分为白盒测试和黑盒测试两种。

（1）白盒测试，又称白箱测试。是指测试人员可以了解系统或者模块的内部结构和处理过程。它根据程序的内部逻辑来设计测试用例，从而可以从内部来检查系统或模块是否都按照设计的要求正确工作。

（2）黑盒测试，又称黑箱测试。与白盒相对的，在黑盒测试中测试人员不需要了解系统或模块的结构和处理过程。它是根据设计规格说明书规定的功能来设计测试用例，主要用来测试系统功能是否符合预期的要求。

11.1.2.4　Web 测试

在单元测试时，可以对系统中某个模块、类或方法进行测试，以便检查它们能否得到预期的结果。与普通系统不同的是，基于 Web 的系统是由页面组成，当需要对基于 Web 的系统进行测试时，需要考虑页面之间的跳转。对此，微软公司也提出了在.NET 下进行 Web 测试的方式。

Web 测试是由一系列的 HTTP 请求组成，它通过发出 HTTP 请求在协议层工作。使用 Web 测试的目的主要有如下几个方面。

（1）对 Web 应用程序的功能进行测试。

（2）对数据驱动进行测试。当 Web 应用程序需要用户提供不同的 Web 页面数据时，用户可以手工输入数据。

（3）对 Web 应用系统的性能进行测试。可用来判断一个基于 Web 的应用系统各项性能指标如何。

Web 测试可自动处理包括隐藏字段相关性、重定向、从属请求和身份验证等 HTTP 操作。进行 Web 测试一般可参照如下步骤进行：

（1）添加 Web 测试。

（2）编辑 Web 测试。

（3）运行 Web 测试并分析结果。

（4）修改测试，再次运行测试。

问题思考

Web 测试是输入 11.1.2.3 小节中 3 个测试步骤中的哪一个步骤呢？

11.1.3 任务实施

1．创建 Web 测试

由于本任务的测试对象为用户登录部分，需要创建针对登录部分的 Web 测试。具体步骤如下：

（1）在 Visual Studio 中新建一个独立的测试项目，命名为“eBuyTest”，如图 11-1 所示。

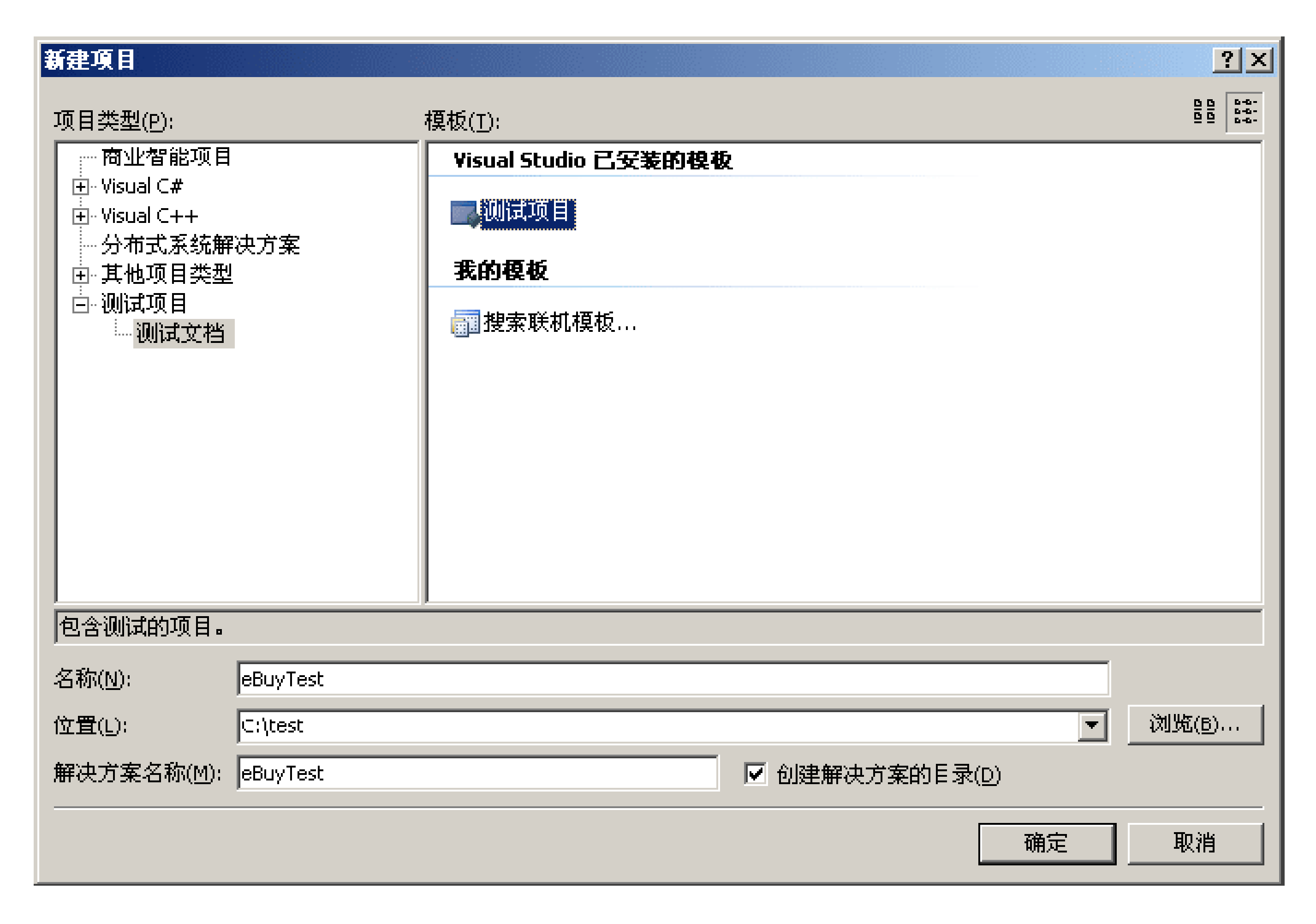

图 11-1 新建测试项目

（2）项目新建好后，可以在“解决方案资源管理器”中看到如图 11-2 所示的内容。

（3）针对测试目标，需要在测试项目中新建一个测试。右击测试项目，在弹出的快捷菜单中选择“添加”|“新建测试”命令，如图 11-3 所示。

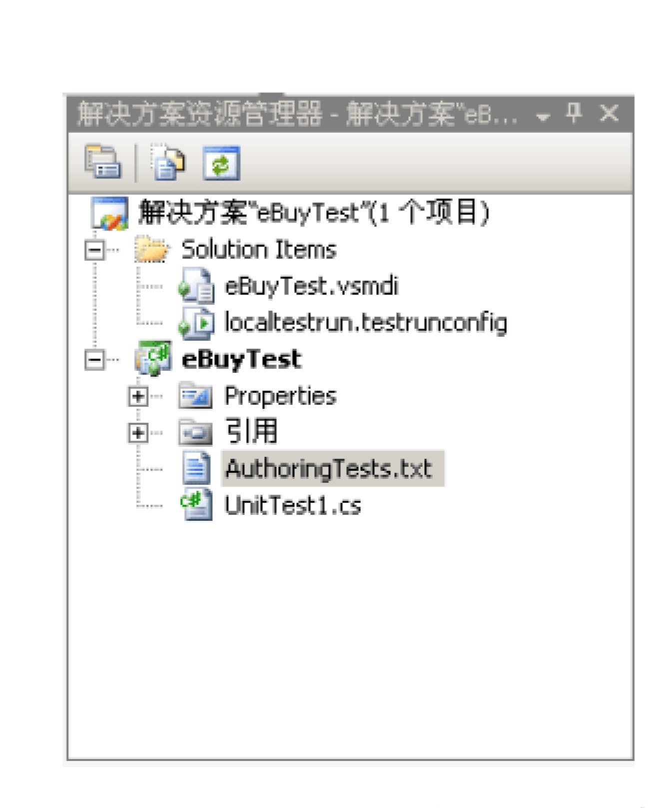

图 11-2 测试项目的解决方案

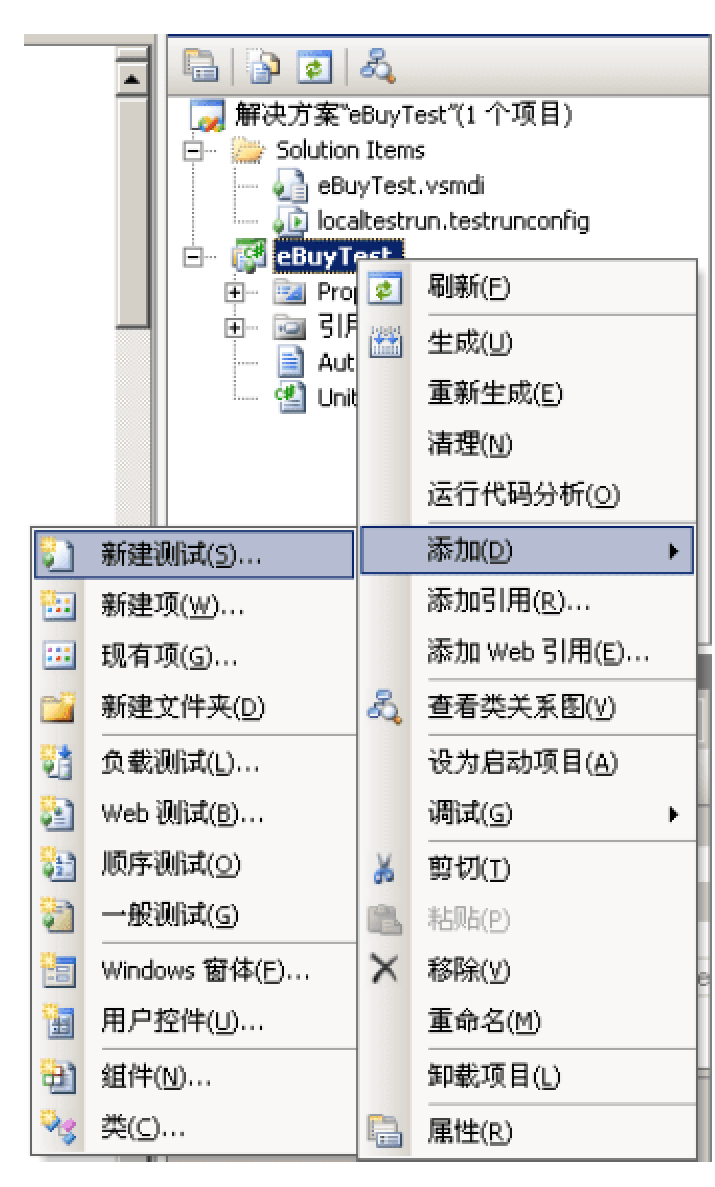

图 11-3 选择“新建测试”命令

（4）在打开的“添加新测试”对话框中选择“Web 测试”选项，并将其命名为“Login.webtest”，表示当前是对登录模块进行测试的，如图 11-4 所示。

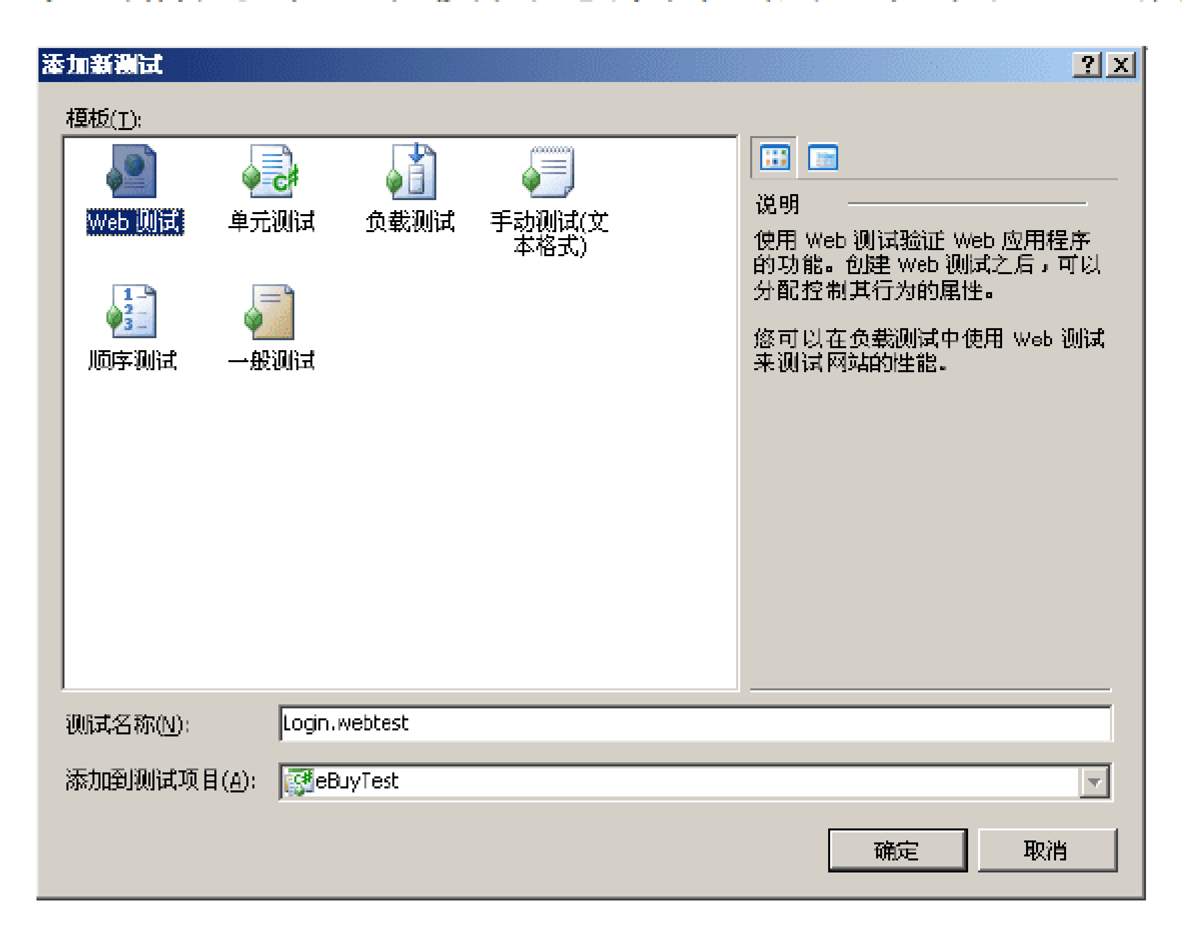

图 11-4　“添加新测试”对话框

（5）单击“确定”按钮即可完成 Web 测试的添加，在解决方案中就可以看到新增加的 Login.webtest 文件。双击此文件，便可在测试文件上添加记录，如图 11-5 所示。

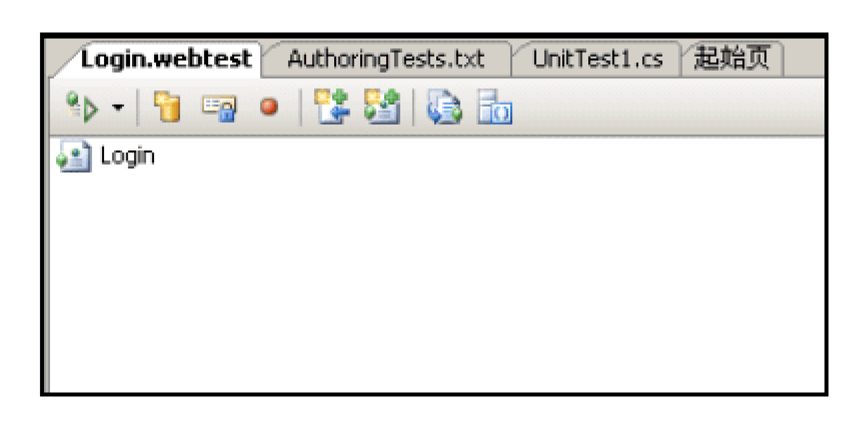

图 11-5　添加完 Web 测试后的解决方案界面

（6）在打开的窗口上方有一个●图标，它是用来添加记录的。单击这个图标后，就会打开一个新的浏览器窗口，测试项目就开始记录测试人员的操作过程，如图 11-6 所示。这时，可以开始按照预先设计好的测试计划在浏览器中输入想要测试的地址，每当访问一个新地址时，左边的测试记录中就会增加一条记录。例如，首先访问了易购商城的首页，那么在左边的记录中就有一条首页地址的记录。

图 11-6　添加记录

（7）如果要测试登录页面，单击页面上的“管理登录”链接后，测试记录又会增加新的一条，如图 11-7 所示。

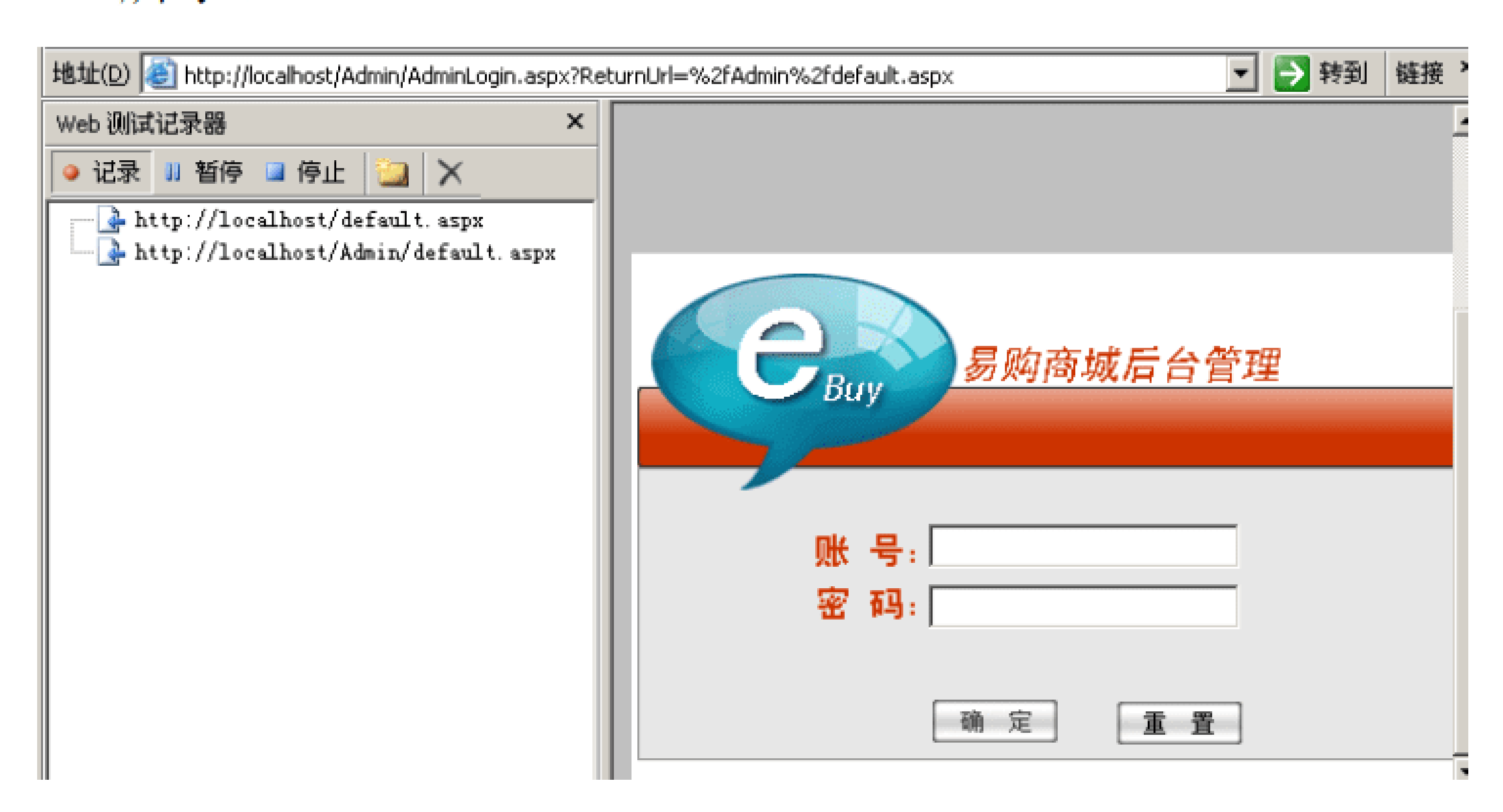

图 11-7 测试管理登录

（8）在如图 11-7 所示的账号和密码文本框中分别输入“admin”和“admin”，成功登录后台管理系统。这时可以看到左边的记录栏发生了一点变化，这次记录的除了地址以外还有两组参数，一组是“QueryString 参数”，另外一组是“窗体发布参数”，里面包含了提交值以及返回值等内容，如图 11-8 所示。

图 11-8 登录后台管理系统

（9）单击左边的“停止”按钮，即可停止测试记录并回到测试编辑器页面，这时在编辑器页面中可以看到刚才记录下来的操作，如图 11-9 所示。

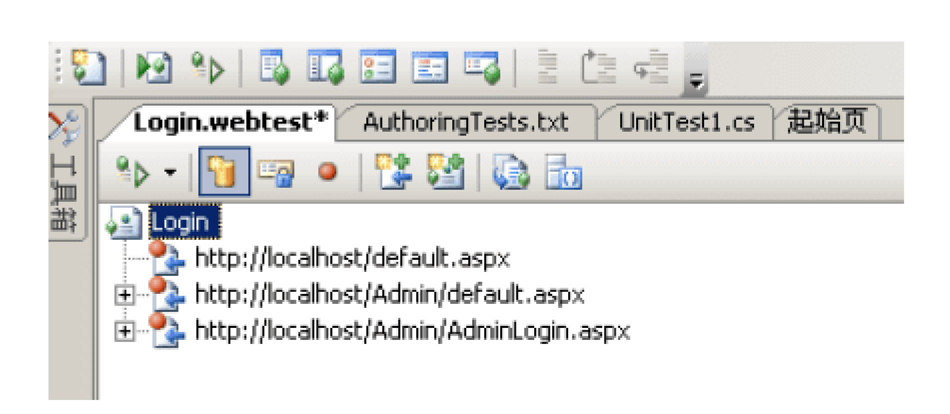

图 11-9 添加的测试记录

2．编辑 Web 测试

（1）在添加好一个 Web 测试以及一些测试记录后，如果需要对从首页登录后台的过程再进行一次测试，只需要直接运行这个测试即可，如图 11-10 所示。

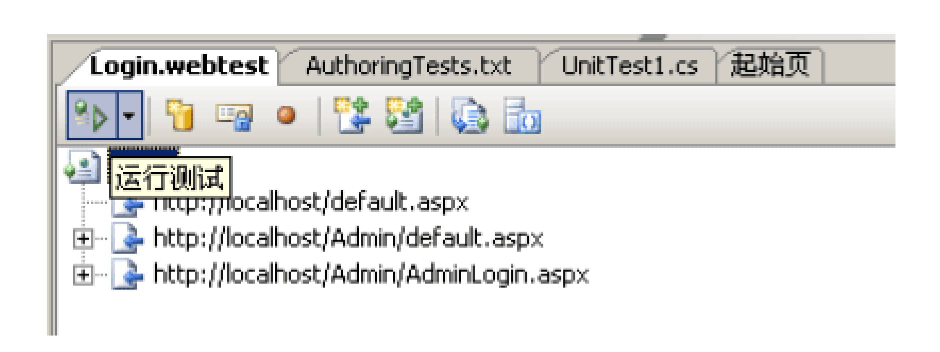

图 11-10　运行测试

（2）运行结束后，可以看到如图 11-11 所示的测试结果。

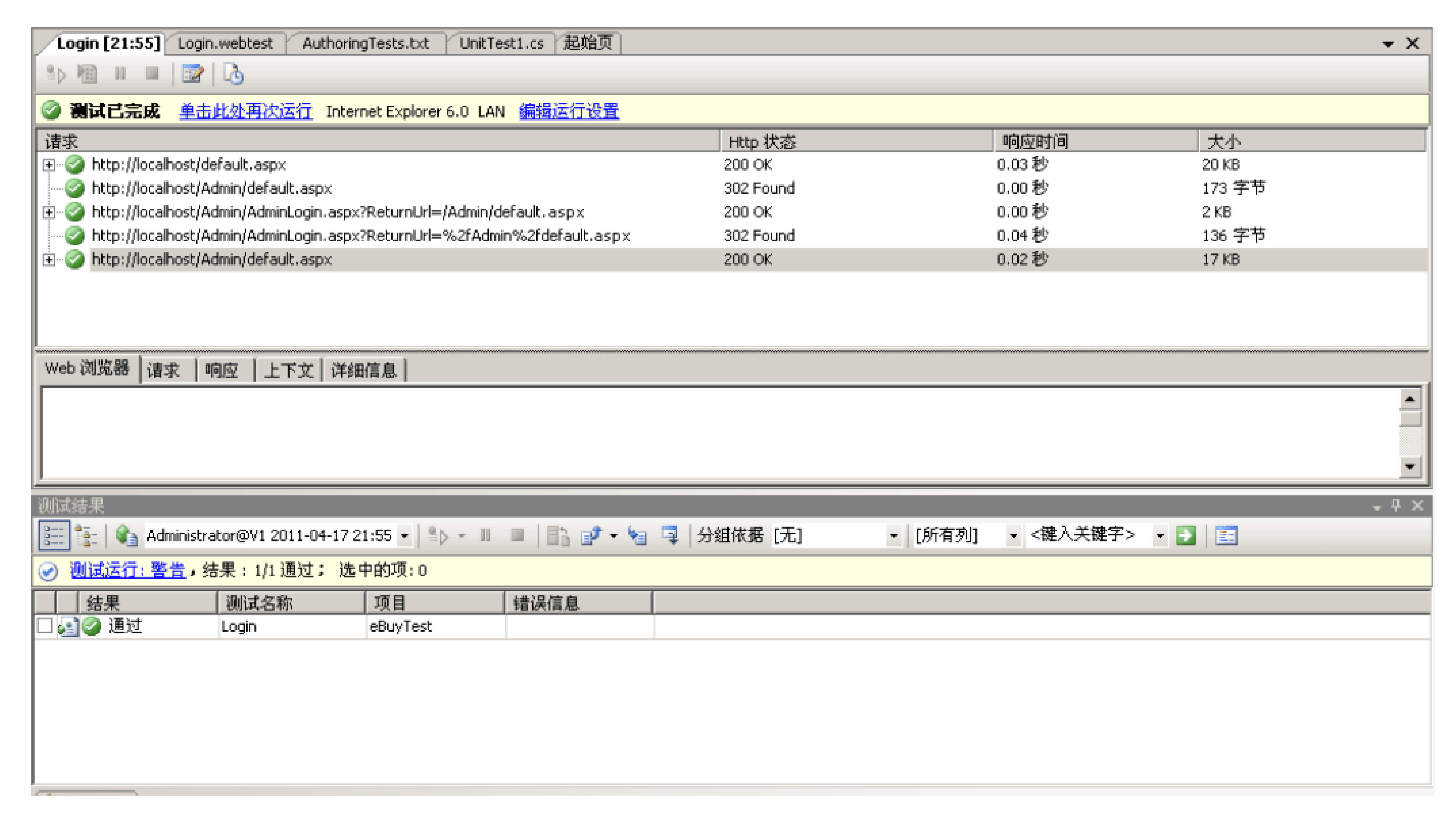

图 11-11　测试结果

（3）不过在更多时候，测试工作不仅仅是停留在简单地将一个测试过程再运行一遍，而是会对测试过程做出修改，以便能够进行一些特定的验证。在 VSTS 中提供了验证规则，通过这些验证规则可以实现不同需求的验证。右击测试记录中的某一项，在弹出的快捷菜单中选择“添加验证规则”命令，如图 11-12 所示。

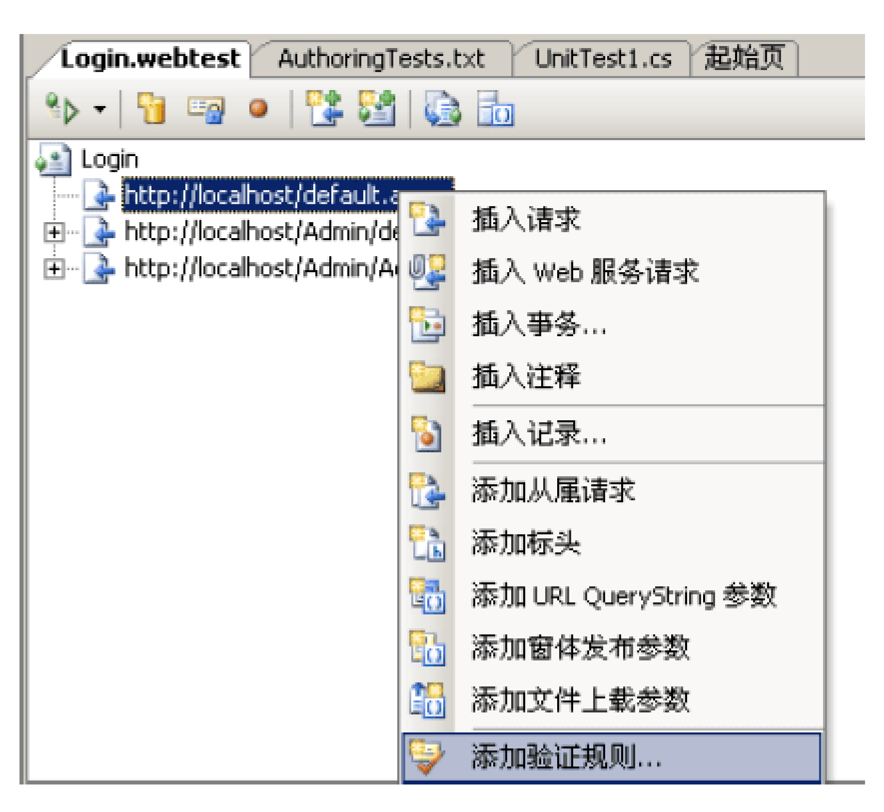

图 11-12　添加验证规则

（4）在打开的“添加验证规则”对话框中，可供选择的验证规则有如下几种。

窗体字段。即验证具有指定名称和值的特定窗体字段是否存在，如图 11-13 所示。

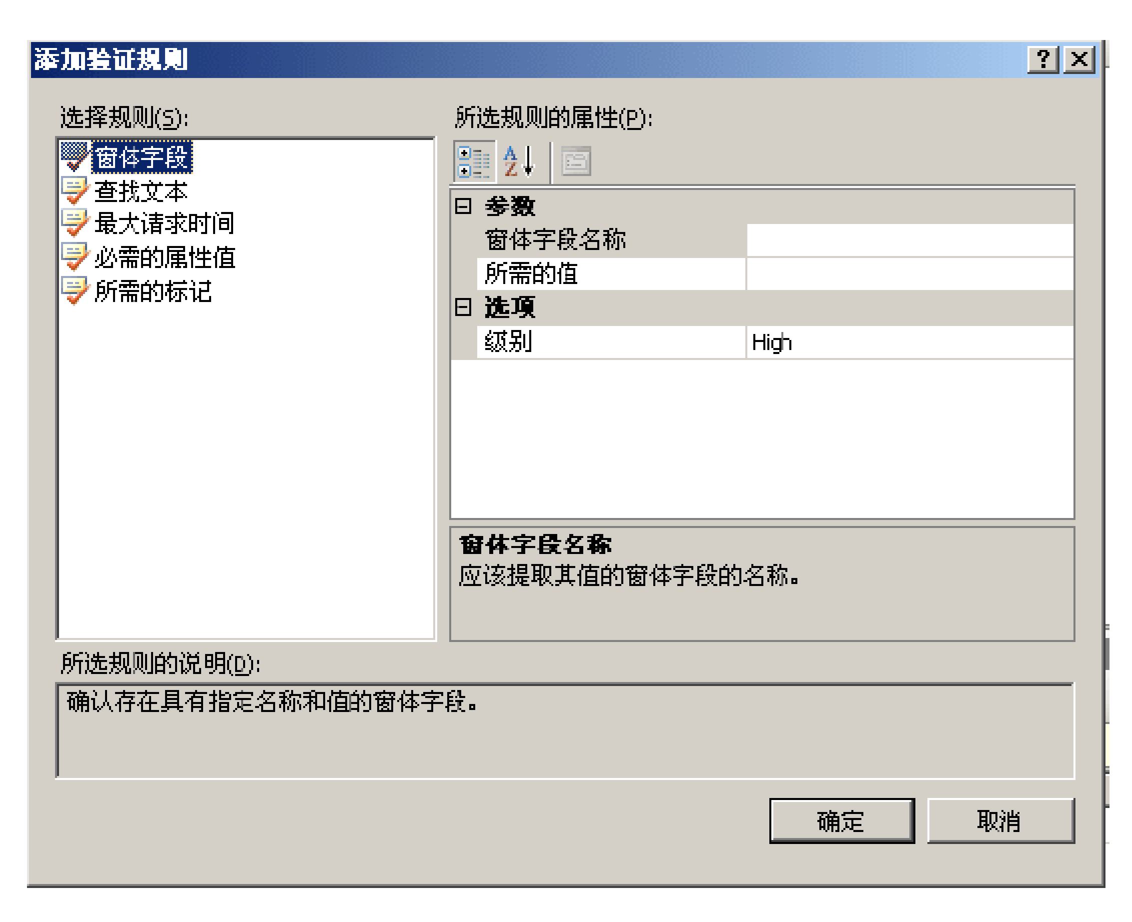

图 11-13 “窗体字段”验证规则

查找文本。即验证返回中是否存在指定的文本，如图 11-14 所示。

图 11-14 “查找文本”验证规则

最大请求时间。即请求是否能在指定的时间内完成。

必需的属性值。即用来验证是否包含具有指定值的属性的 HTML 标记所需的标记。即用来验证是否存在指定的 HTML 标记（如图 11-15 所示）。

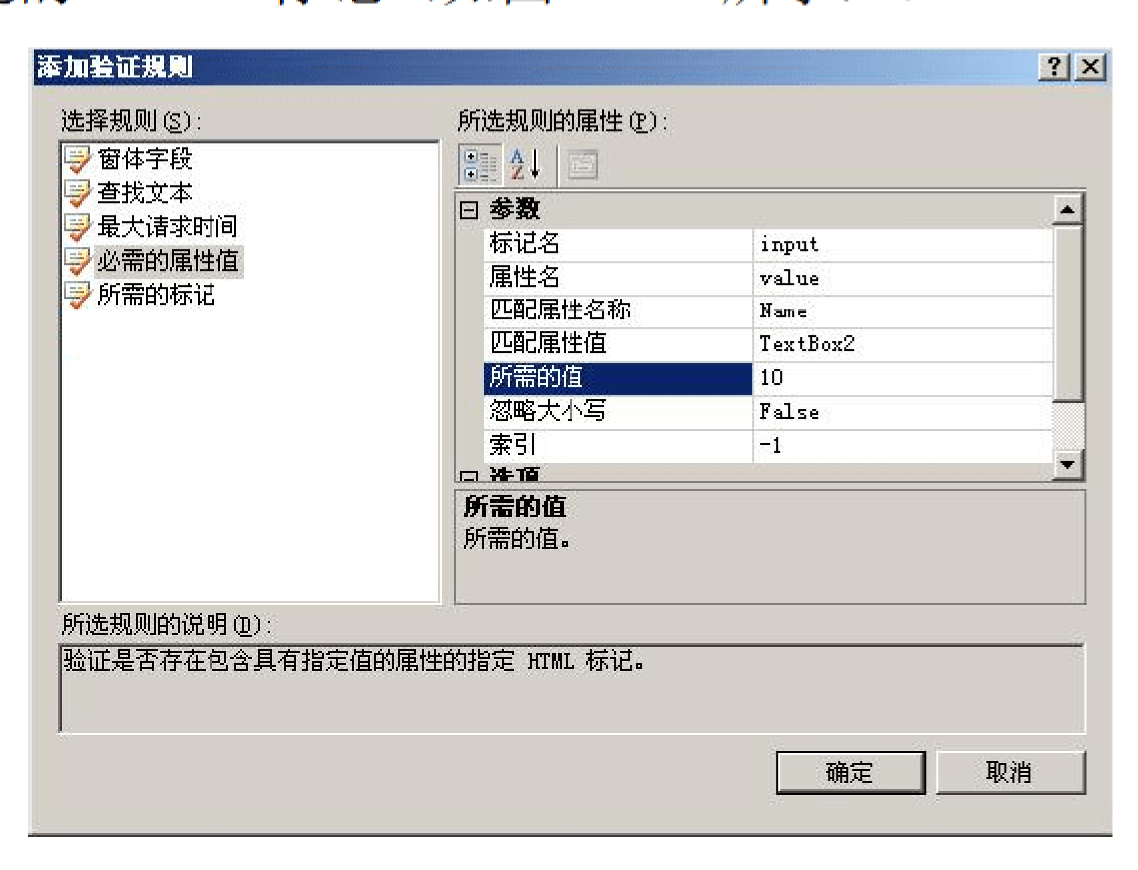

图 11-15 “必需的属性值”验证规则

（5）接下来就对 3 个测试记录分别添加如下的验证规则：给第 1 条记录添加最大响应时间规则，要求最大响应时间不超过 100 毫秒；给第 2 条记录添加窗体字段规则，要求查找的窗体字段的名称为“txtLoginId”，值为空；给第 3 条记录添加查找文本规则，要求查找文本的内容为“admin”。添加验证规则后的页面内容如图 11-16 所示。

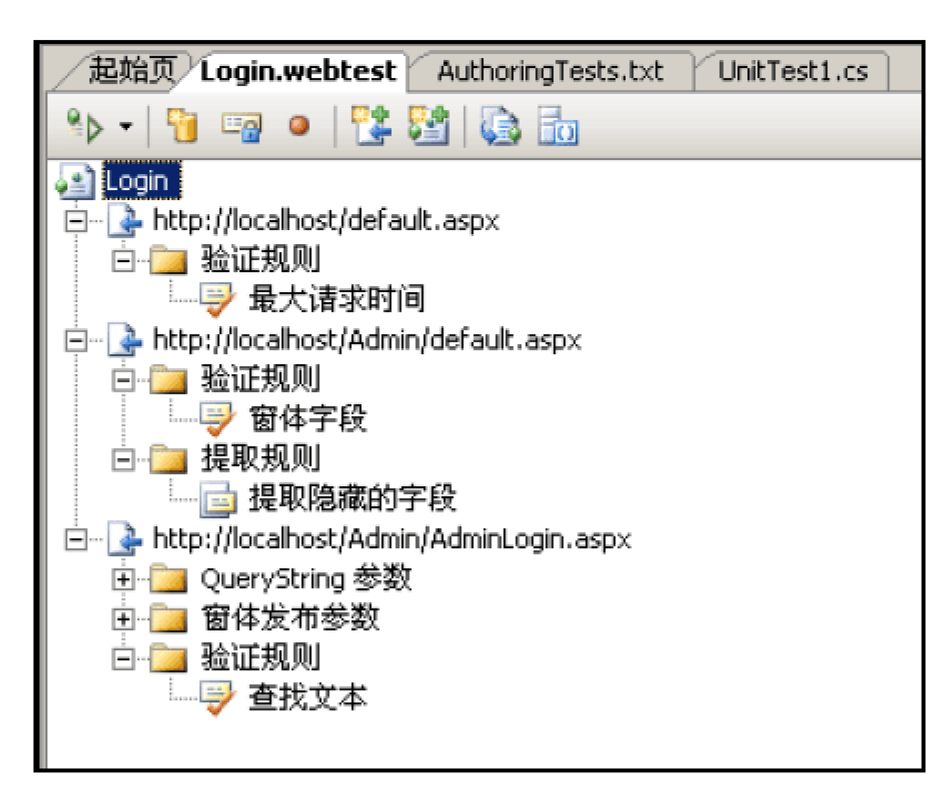

图 11-16　添加验证规则后的页面内容

3．运行 Web 测试并分析结果

（1）添加好指定的验证规则后就可以运行测试了，单击“运行测试”按钮即可。运行完成后，可以看到如图 11-17 所示的测试结果。

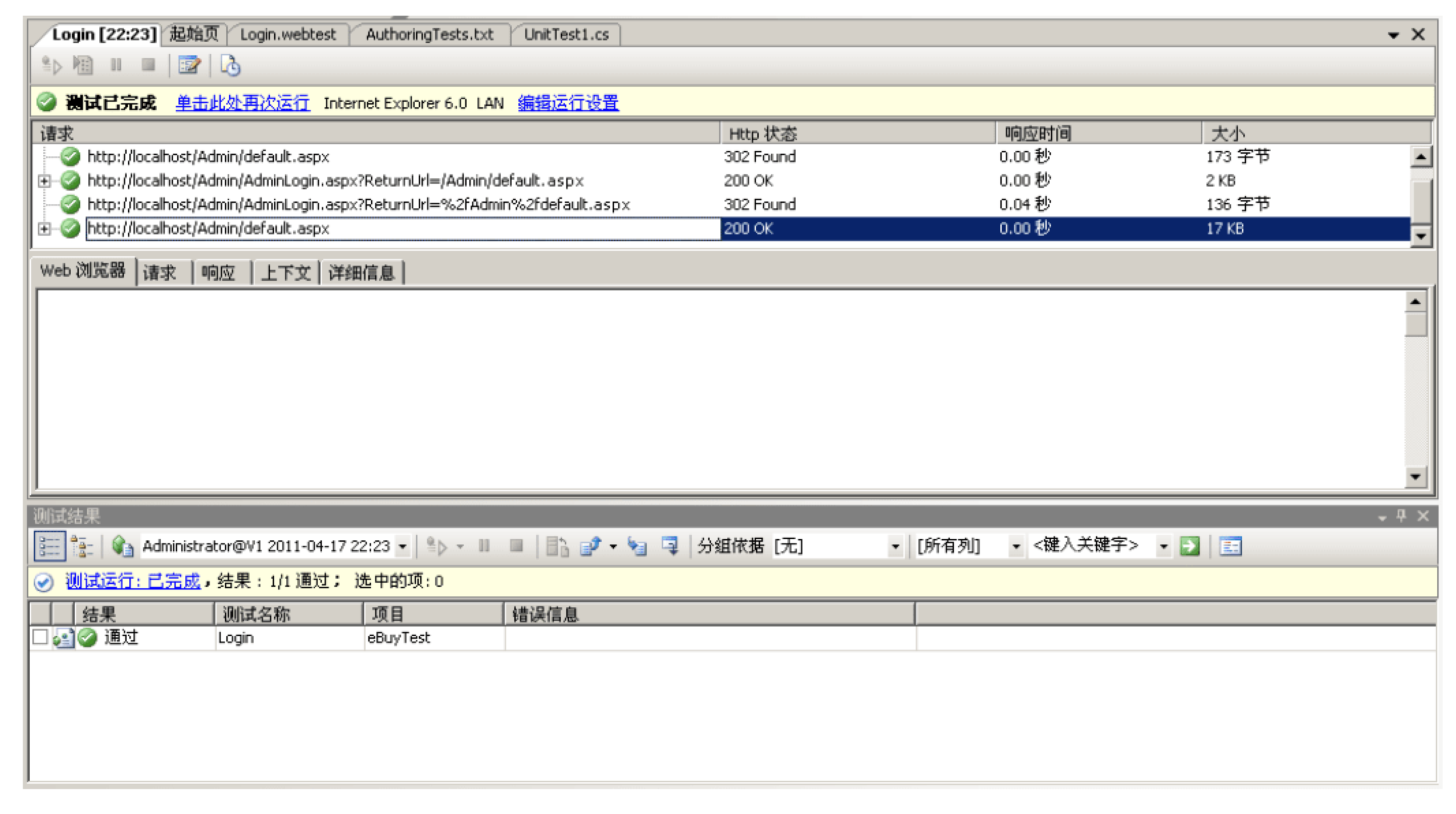

图 11-17　测试结果

（2）测试的结果为“通过”，表示先前设置的验证规则都成功了。可以通过查看详细信息来分析测试过程中的每个验证规则。例如要查看第 3 条记录设置的查找文本验证规则，可以单击记录后查看详细信息，如图 11-18 所示。

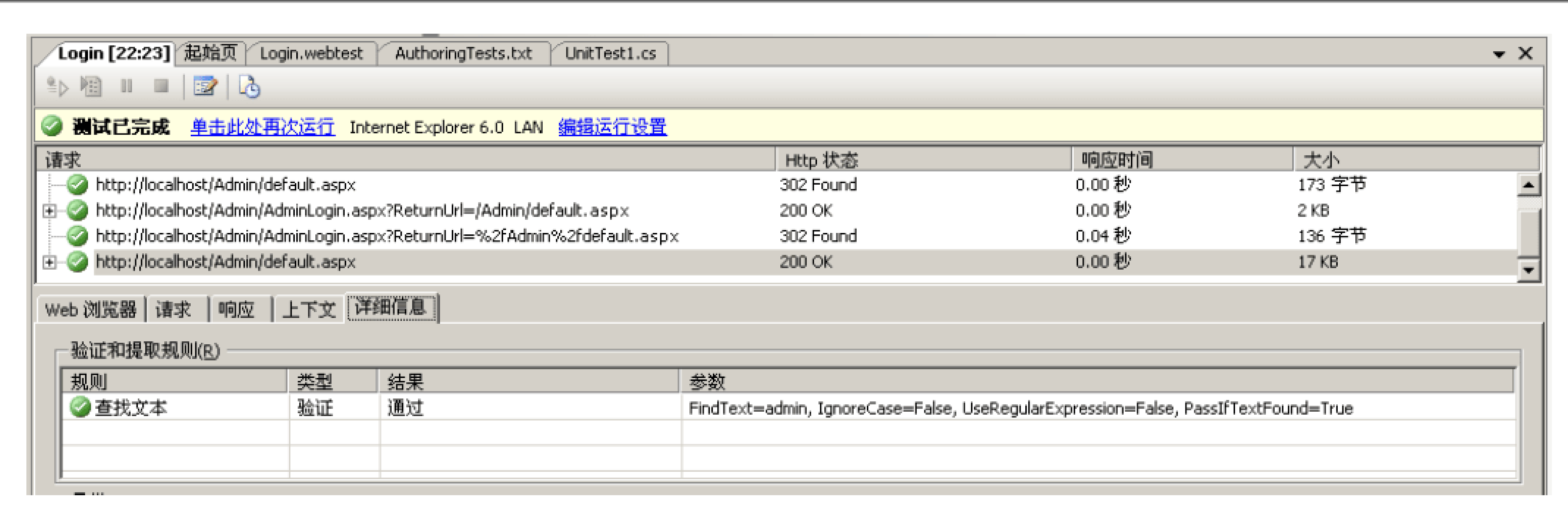

图 11-18　查看验证规则的详细信息

4．修改测试并再运行测试

现在将第 1 条记录的最大响应时间从 100 毫秒改为 10 毫秒，看看会发生什么情况。运行修改后的测试，可以看到如图 11-19 所示的测试结果。这时，测试结果为失败。

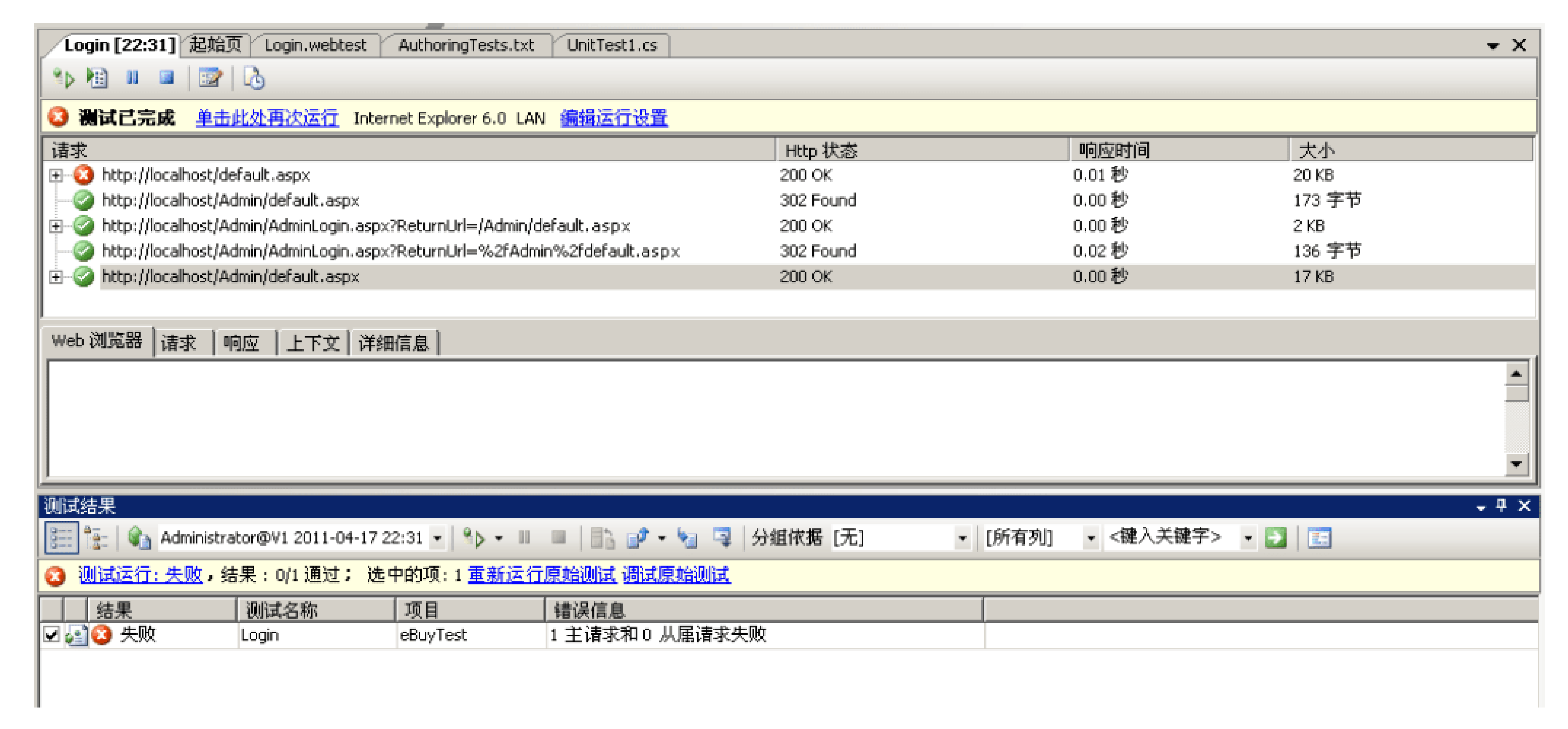

图 11-19　测试结果

是什么原因导致修改响应时间后的测试失败呢？可以通过分析测试结果来找到答案。查看失败记录的详细信息，可以看到如图 11-20 所示的内容。从中可以看到，第 1 条测试记录的真实响应时间为 13.5288 毫秒，而设置的验证规则时间为 10 毫秒，比真实响应时间少，因此测试失败。

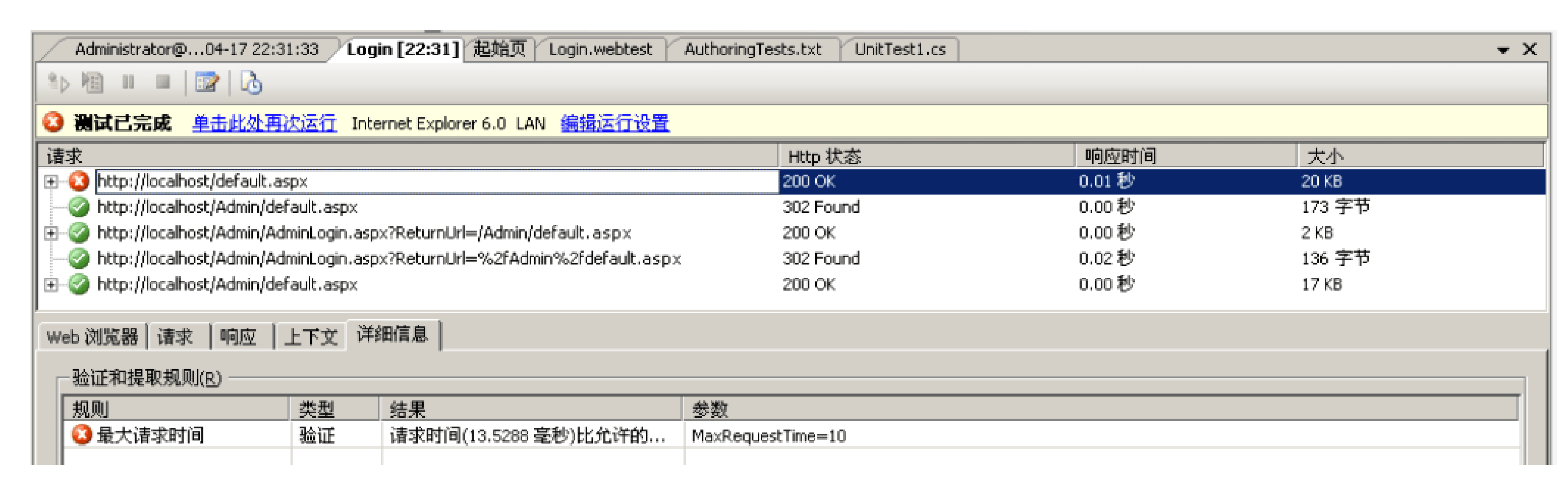

图 11-20　失败记录的详细信息

通过这种方式就可以按照预先的要求来设计并执行测试，分析测试的结果可以找到系统可能存在的问题。

11.1.4　任务评价

标　准	掌 握 情 况
能使用 VSTS 创建测试	熟练□　基本□　模糊□
能适当地设计验证规则来实现测试预期	熟练□　基本□　模糊□
能分析测试结果找到存在的问题	熟练□　基本□　模糊□
自 我 小 结	

11.1.5　任务拓展

参照本节的内容，读者可以采用这种测试的方法对用户注册部分的页面进行 Web 测试，并通过调整测试参数来对测试结果进行分析。

任务 11.2　对系统进行负载测试

11.2.1　任务分析

任务目标：为了测试系统的性能，还需要对实际使用环境中发生的访问进行模拟，从而检验系统是否能够在一定的范围内正常工作。在这个任务中，将使用 VSTS 的负载测试工具对系统的商品信息查询模块进行负载测试，以分析该模块在不同负载情况下的性能反应，为以后系统的优化做参考。

完成标准：对商品信息查询模块进行负载测试，并记录测试结果。

应用手段：使用 VSTS 的负载测试工具来进行。

11.2.2　相关知识

11.2.2.1　负载测试

负载测试（Load testing）是指通过测试系统在资源超负荷情况下的表现，来发现系统设计上的错误或验证系统的负载能力。在这种测试中，将使测试对象承担不同的工作量，

以评测和评估测试对象在不同工作量的条件下的性能行为，以及持续正常运行的能力。负载测试的目标是确定并确保系统在超出最大预期工作量的情况下仍能正常运行。此外，负载测试还要评估性能特征。例如，响应时间、事务处理速率和其他与时间相关的方面。

负载测试是模拟系统在实际环境下所承受的负载条件的系统负荷，通过不断加载（如逐渐增加模拟用户的数量）或其他加载方式来观察不同负载下系统的响应时间和数据吞吐量、系统占用的资源（如 CPU、内存等）等，以检验系统的行为和特性，从而发现系统可能存在的性能瓶颈、内存泄漏、不能实时同步等问题。负载测试需要通过系统性能特性或行为来发现系统的性能问题，从而为性能改进提供帮助。

知识点小贴士

通常与负载测试有关的测试技术有：性能测试、负载测试以及压力测试等。

性能测试是指通过自动化的测试工具模拟多种正常、峰值以及异常负载条件来对系统的各项性能指标进行测试。

负载测试是指确定在各种工作负载下系统的性能，目标是测试当负载逐渐增加时，系统各项性能指标的变化情况。

压力测试是指通过确定一个系统的瓶颈或者不能接收的性能点，来获得系统能提供的最大服务级别的测试。

11.2.2.2 负载测试工具

负载测试一般需要专门的工具来进行，主流的商用负载测试工具有如下的几种。

1. LoadRunner

LoadRunner 是由 HP 公司开发的一款较高规模适应性的、自动负载测试工具。这个工具可以预测系统行为，优化性能。LoadRunner 强调的是整个企业的系统，通过模拟实际用户的操作行为和实行实时性能监测，来帮助开发者更快地确认和查找问题。同时，LoadRunner 能支持最宽泛的协议和技术，为用户的特殊环境，提供解决方案。

2. QALoader

QALoader 是 Compuware 公司开发的一套针对客户/服务器系统、企业资源配置（ERP）和电子商务应用的自动化负载测试工具。QALoad 是 QACenter 性能版的一部分，它通过可重复的、真实的测试能够彻底地度量应用的可扩展性和性能。

3. WebRunner

WebRunner 是一款由 Rad View 公司推出的性能测试和分析工具。它的主要功能是允许 Web 应用程序开发者自动执行压力测试。WebRunner 通过模拟真实用户的操作，生成压力负载来测试 Web 的性能，用户创建基于 JavaScript 的测试脚本，称为议程 agenda，用它来模拟客户的行为，通过执行该脚本来衡量 Web 应用程序在真实环境下的性能。

以上 3 种负载测试工具之间的比较如表 11-1 所示。

表 11-1　主流的商用负载测试工具

名　　称	LoadRunner	QALoader	WebRunner
出品公司	HP(Mercury)	Compuware	Radview
价格	昂贵	较贵	一般
安装配置的复杂性	简单	简单	一般
操作性	较复杂	简单	简单
支持测试对象	各种中间件/数据库/应用服务器的性能监控/企业架构（j2ee 和.net）的测试	客户/服务器系统、企业资源配置（ERP）和电子商务应用	Web Application
支持平台	windows,unix 或 linux	HP-UX, IBM AIX,Sun Solaris, Linux, NT/2k	Unix Windows
支持数据库	DB2,SQLserver,Orcale, Sybase	ADO, DB2,Oracle,Sybase, SQLserver,Odbc	ADO,DB2,Oracle,Sybase, SQLserver,Odbc
支持协议	web,http(s),soap,streaming, wap,winsock,xml	http,ssl,soap,xml,streaming, media	xml,java,ejb, activex,wap,http,snmp, real/m$streaming
脚本语言	类似 C++	C/C++和 VC++	Javascript
自动数据生成	Y	Y	Y
脚本调试	Y	Y	Y
报表定制功能	Y	Y	Y
功能点	创建虚拟用户，创建真实的负载，定位性能问题，分析结果以精确定位问题所在，重复测试保证系统发布的高性能等	预测系统性能、通过重复测试寻找瓶颈问题、从控制中心管理全局负载测试、快速创建仿真的测试、验证应用的可扩展性	强大的专业网站性能测试，虚拟多用户
虚拟用户上限数量	成千上万	成百上千	理论上无限，不过受机器的限制，同时运行太多影响结果的准确性

除了这些主流的商品负载测试工具之外，还有一些如微软的 WAS（Web Application Stress Tool）以及 OpenSTA 等免费的负载测试工具。在任务中，将采用 VSTS 的负载测试工具对系统进行负载测试，使用时一般按照如下的步骤来进行。

（1）创建负载测试。

（2）运行负载测试。

（3）分析负载测试结果。

11.2.3　任务实施

1．创建负载测试

（1）在进行商品查询模块的负载测试前，首先需要在已有的测试项目中添加一个名

为“SearchTest”的 Web 测试，添加 Web 测试的方法大家可以参照 11.1 节来完成。

（2）商品查询的 Web 测试添加好后，就可以开始对这个模块进行负载测试。右击测试项目，在弹出的快捷菜单中选择“添加”|“新建测试”命令，如图 11-21 所示。

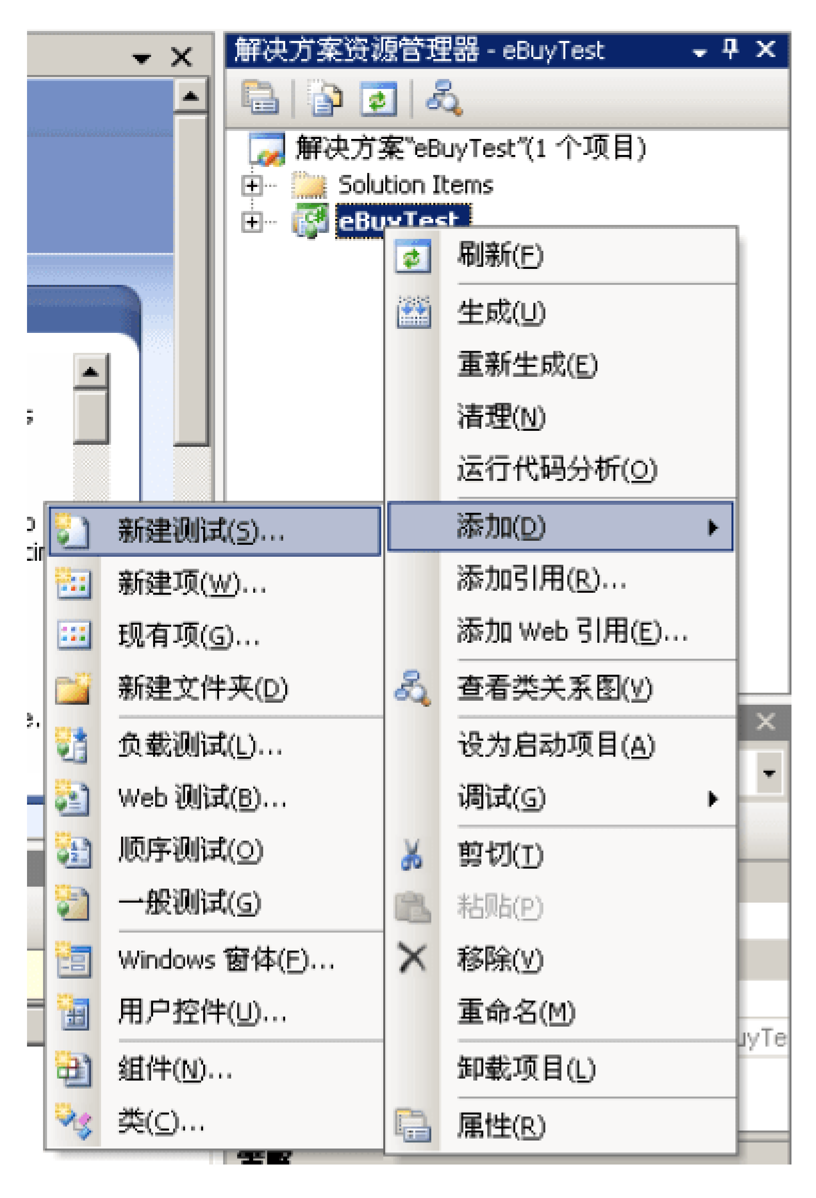

图 11-21　选择“新建测试”命令

（3）在打开的“添加新测试”对话框中，选择“负载测试”选项，将测试名称改为“SearchLoadTest”，如图 11-22 所示。

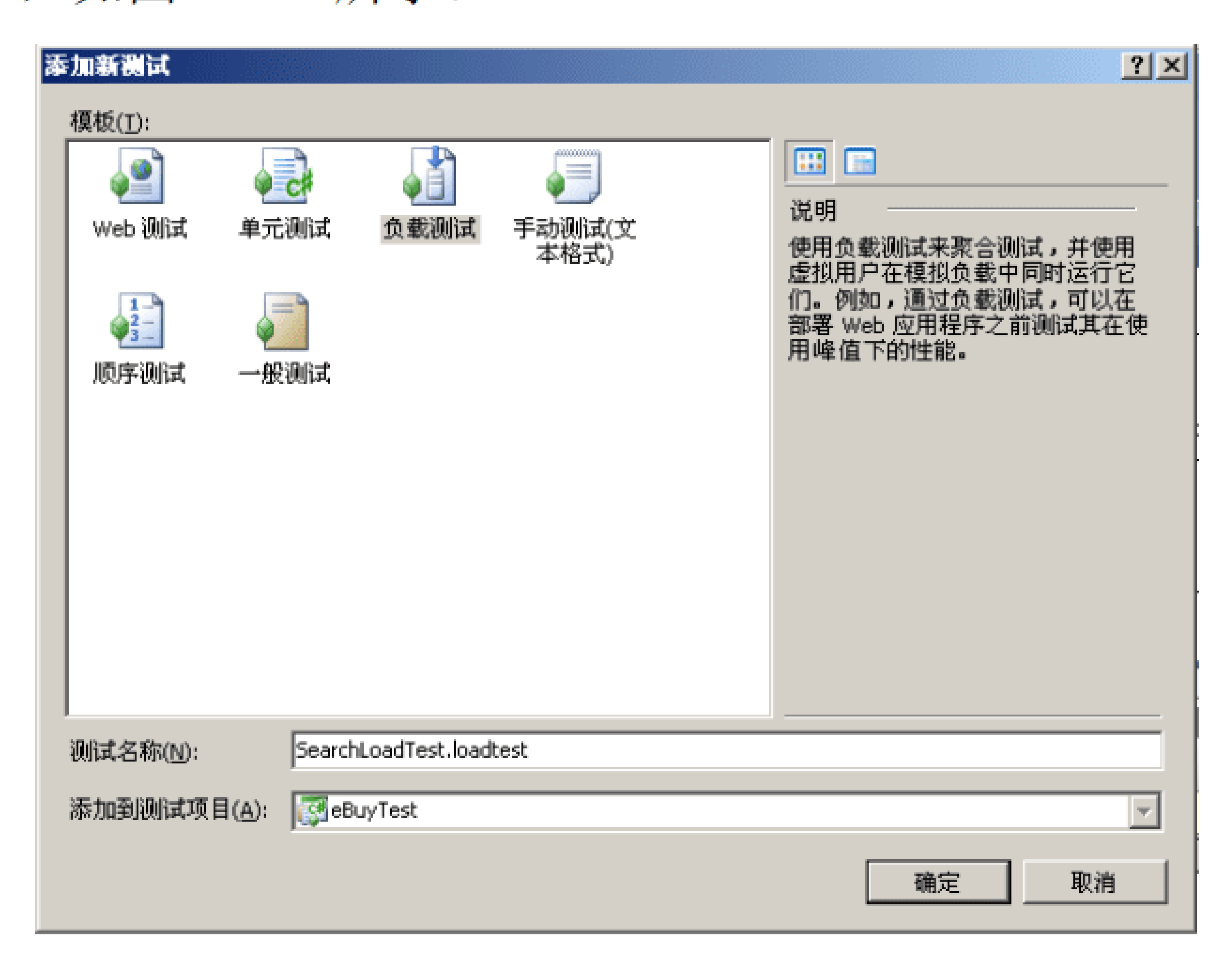

图 11-22　“添加新测试”对话框

（4）单击“确定”按钮，打开“新建负载测试向导”界面，如图 11-23 所示。负载测试向导将会引导测试者进行负载测试的相关设置。

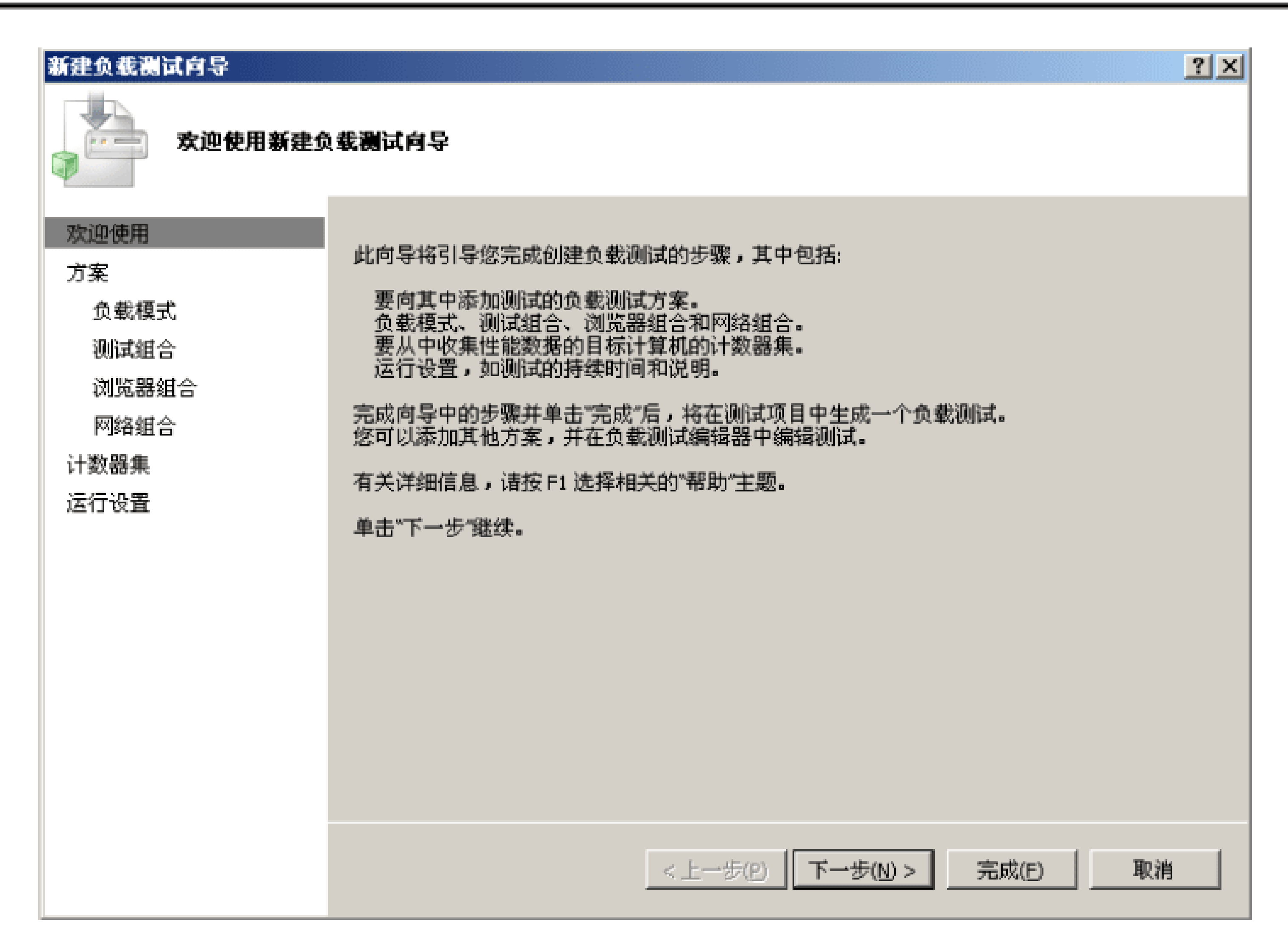

图 11-23 “新建负载测试向导”对话框

（5）单击“下一步”按钮，即可进入如图 11-24 所示的“编辑负载测试方案的设置”界面。在这个界面中，可以设置负载测试方案的名称、思考时间配置文件以及测试迭代之间的思考时间。这里的思考时间是指用户在进行思考而未与系统进行交互时系统等待的时间。而测试迭代之间的思考时间就是模拟测试方案的各测试迭代之间产生的思考时间。

图 11-24 “编辑负载测试方案的设置”对话框

（6）单击“下一步”按钮，进入如图 11-25 所示的“负载模式”界面，这个步骤是用来设置模拟系统的并发访问人数。负载模式分为两种：常量负载和分级负载。其中常量负载是指在整个负载测试过程中模拟的并发访问用户数量是固定不变的。而分级负载是指在测试过程中并发用户数按照设定的值进行变化。在此，选择常量负载模式并将模拟的用户数设置为 100。

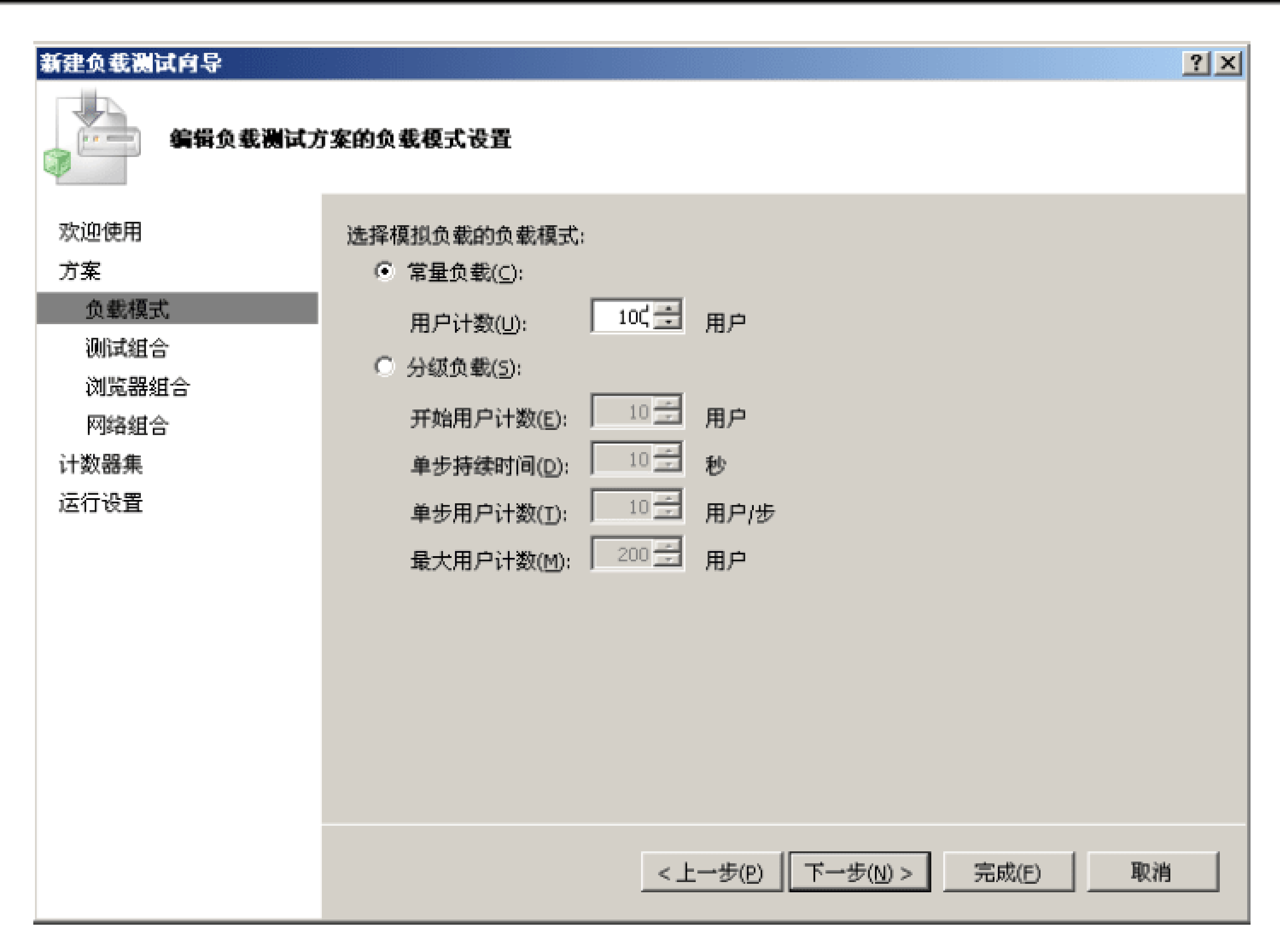

图 11-25 “负载模式”对话框

（7）单击“下一步”按钮，进入如图 11-26 所示的“组合测试编辑”界面。组合测试是指可以将一个或多个测试任务组合在一起进行测试。例如可以将用户注册、用户登录两个功能模块组合在一起进行测试，也可以将上一个任务中的用户登录和这个任务中的商品搜索放在一起进行测试。

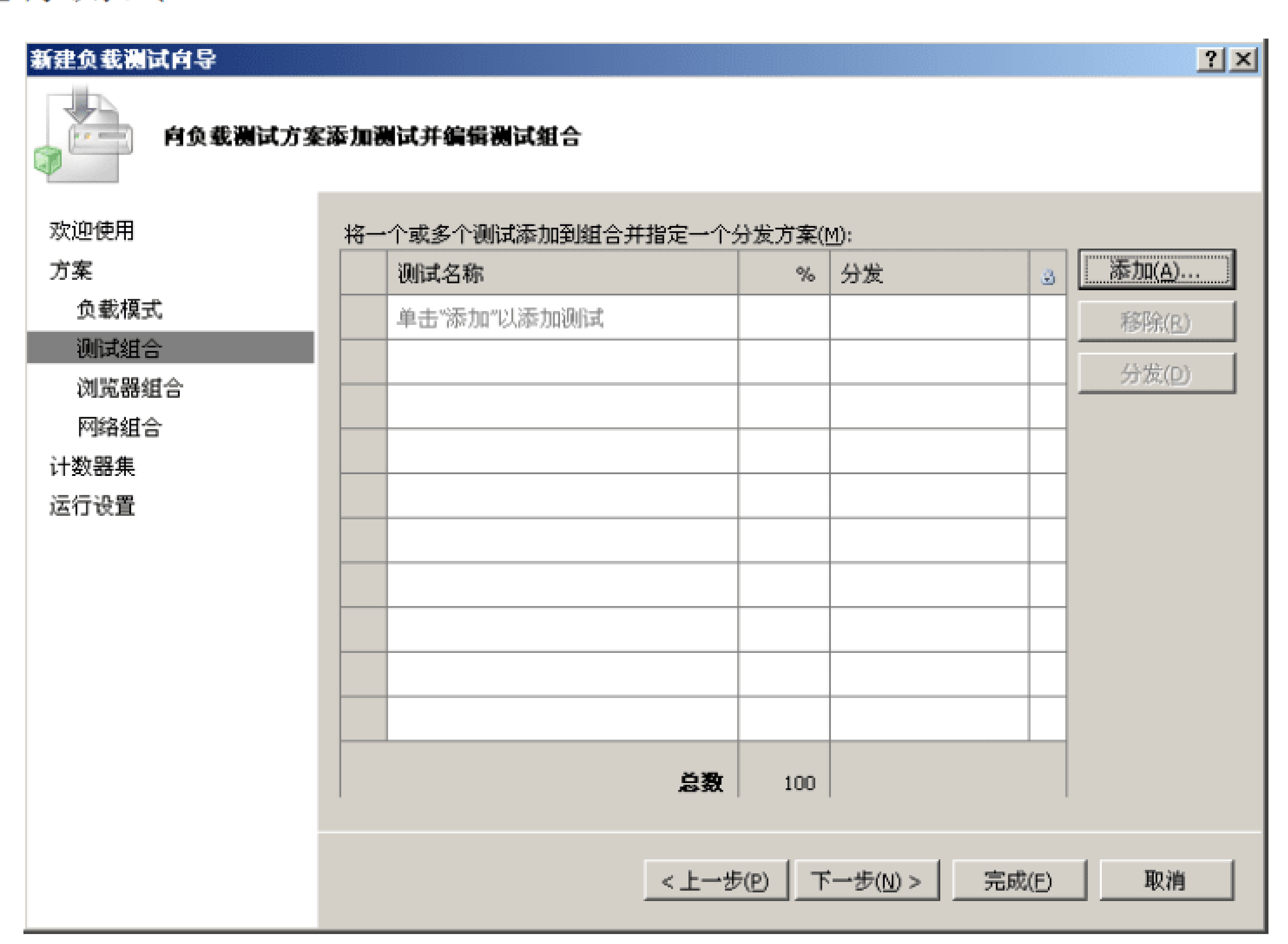

图 11-26 设置测试组合

（8）单击“添加”按钮，可以打开如图 11-27 所示的“添加测试”对话框，可以看到当前测试项目中已有的测试，在此选择 SearchTest 测试项，并将它添加到右边的选定测试中。

（9）单击“下一步”按钮，进入“浏览器组合”编辑界面。这一步中主要模拟用户通过不同的浏览器来访问系统，并可以指定各种浏览器使用的比例。在此全部采用 IE 浏览器来进行模拟，如图 11-28 所示。

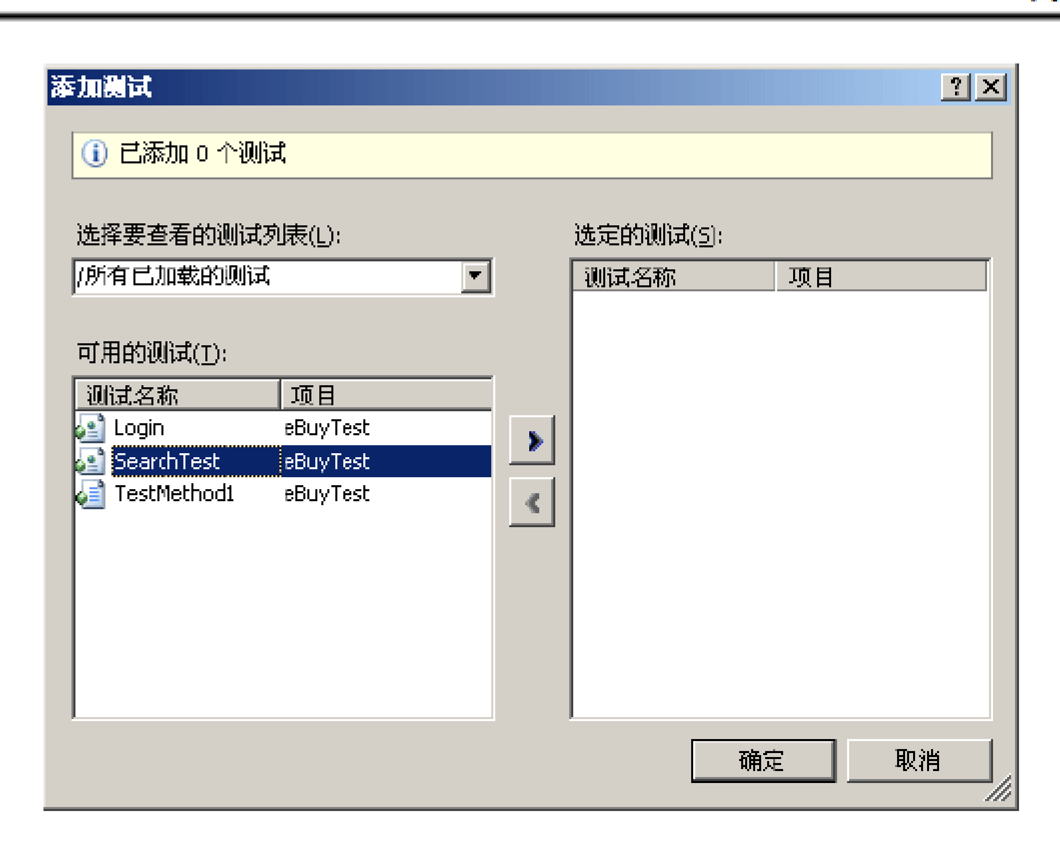

图 11-27　选择测试项

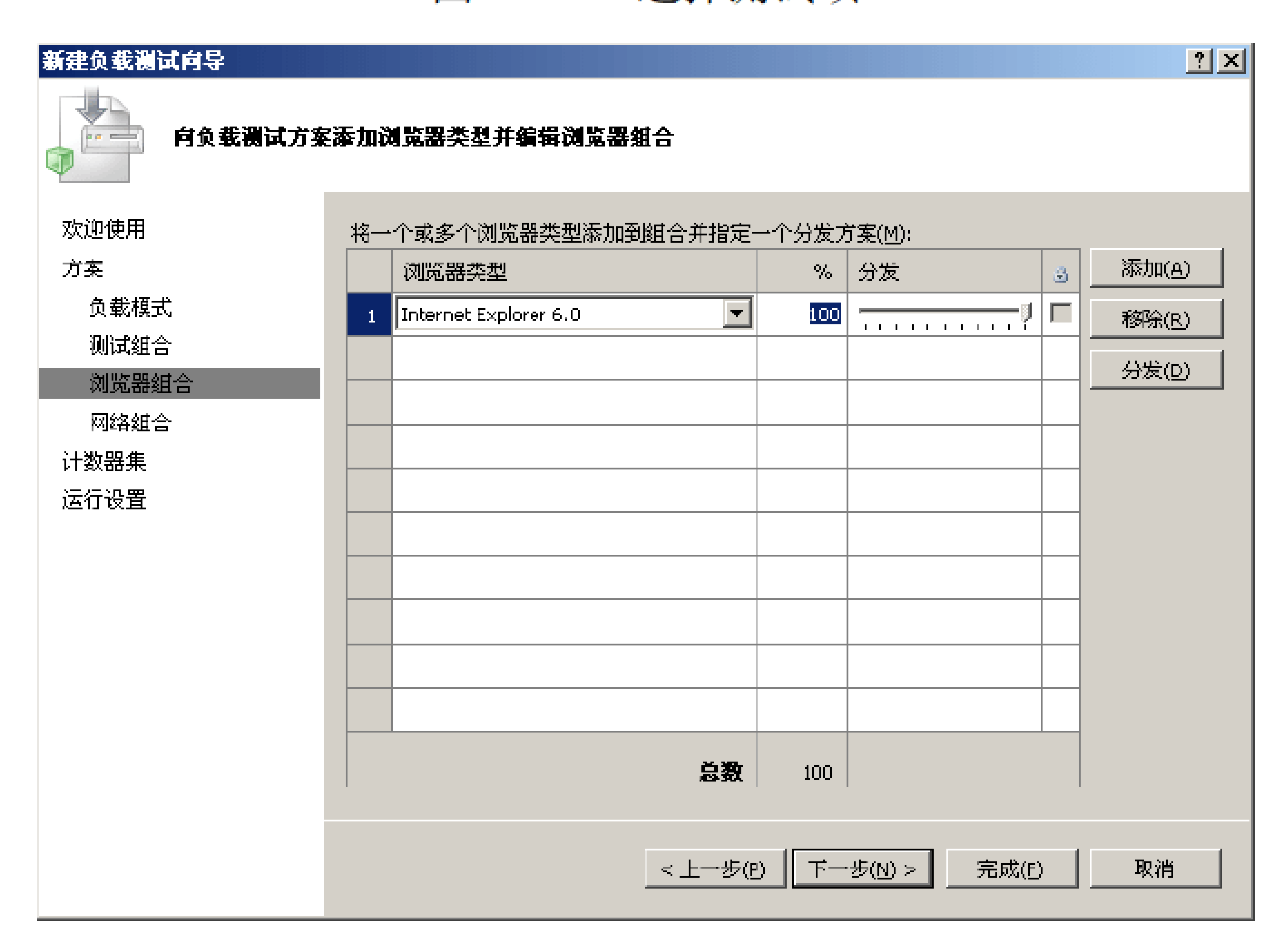

图 11-28　设置浏览器组合

（10）单击“下一步”按钮，进入“网络组合”编辑界面，在这一步中可以模拟不同的网络环境来测试系统的响应情况，如图 11-29 所示。

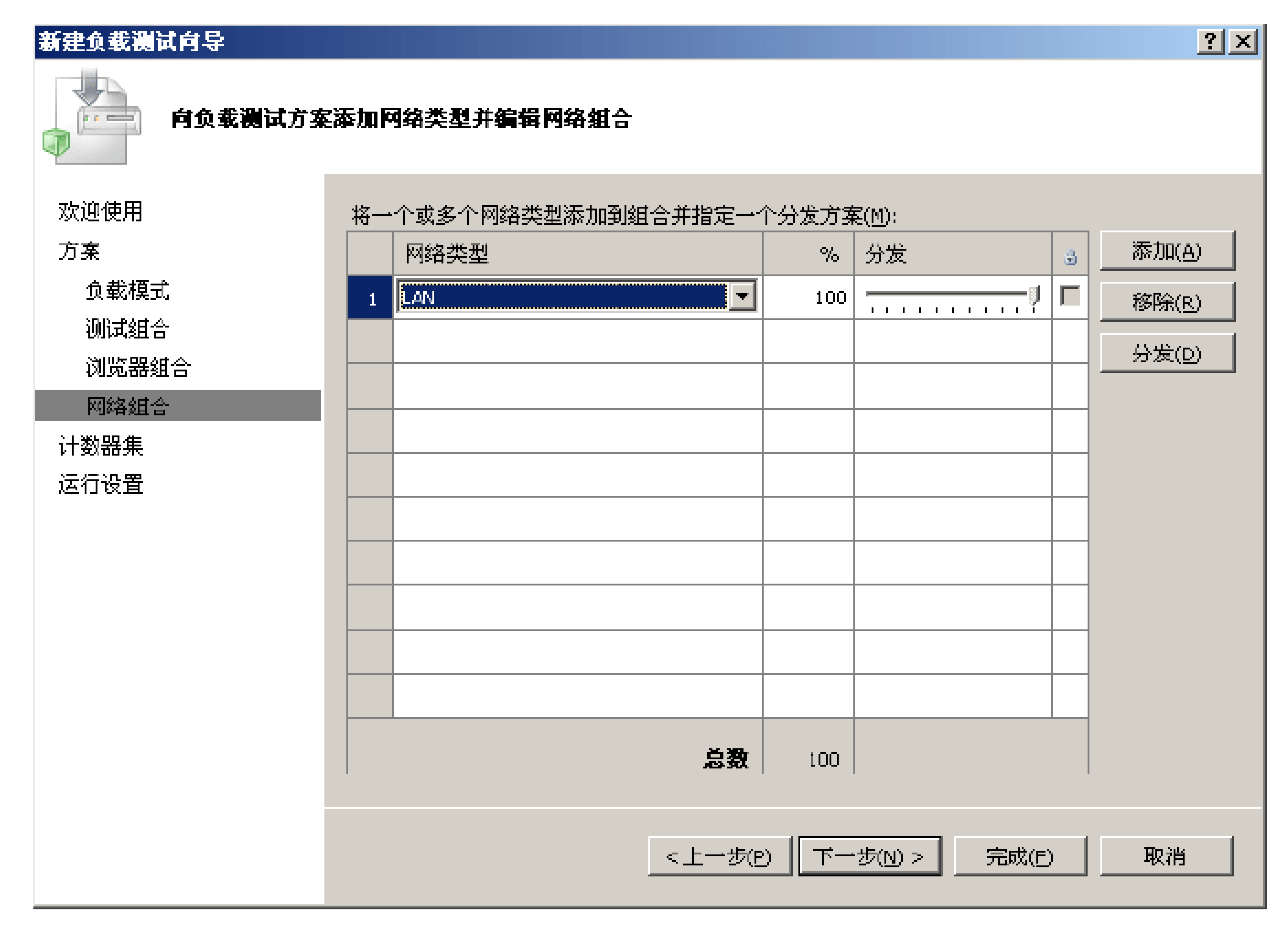

图 11-29　设置网络组合

（11）单击“下一步”按钮，进入“计数器集”界面。通过计数器集的相关设置，可以用更多的计算机来分担模拟数很大时候的服务器压力，如图 11-30 所示。

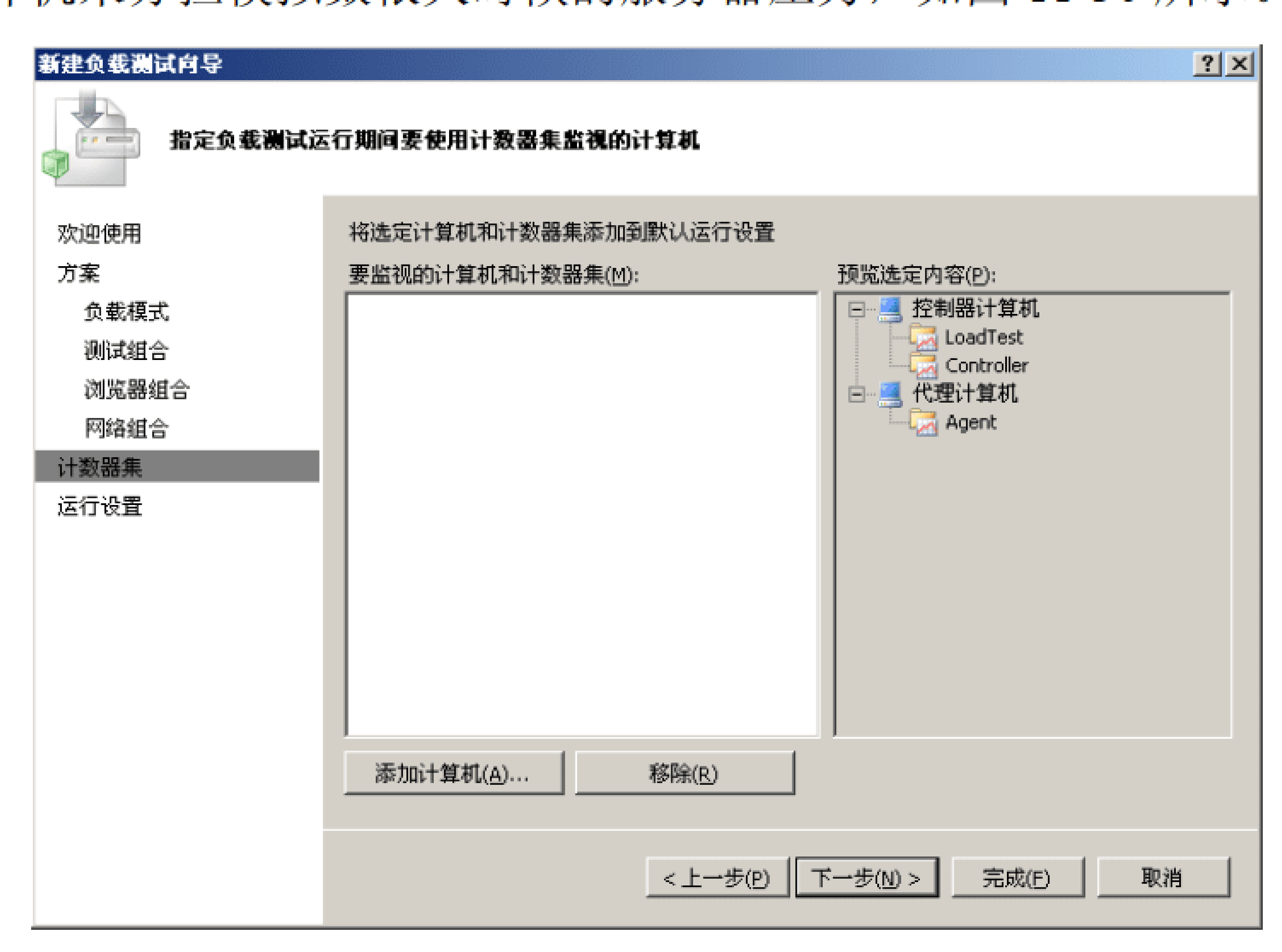

图 11-30　设置计数器集

（12）单击“下一步”按钮，进入“运行设置”界面，此界面主要包括：预热持续时间、运行持续时间以及采样速率。预热持续时间是指从测试开始到开始记录数据之间的间隔时间。运行持续时间是指整个测试的时间长度，负载测试会在指定的运行持续时间内按之前的设置进行用户访问模拟。采样速率是指捕获计数器值的时间间隔，间隔时间越短，捕捉的频率越高，数据越准确。在这里将预热时间设置为 10 秒，持续时间设置为 1 分钟，采样间隔时间设置为 5 秒，如图 11-31 所示。

图 11-31　设置执行时间

（13）单击“完成”按钮，结束负载测试的设置。这时，可以看到如图 11-32 所示的负载测试项目的配置信息。

2．运行负载测试

打开负载测试后，单击“运行测试”按钮即可开始运行负载测试，如图 11-33 所示，然后等待测试结束。

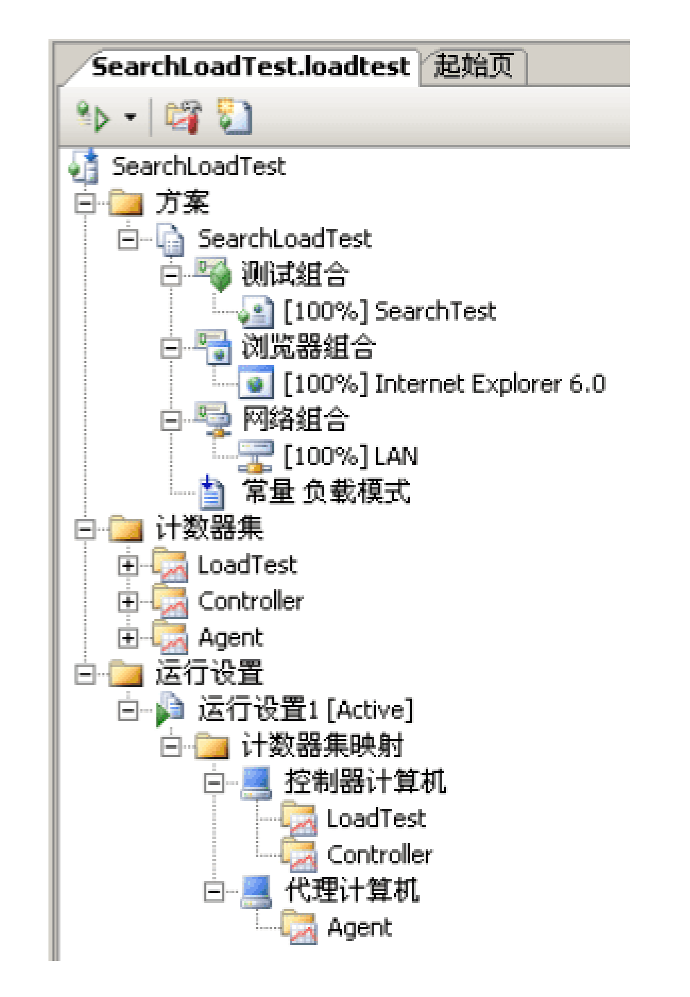

图 11-32　负载测试的配置信息

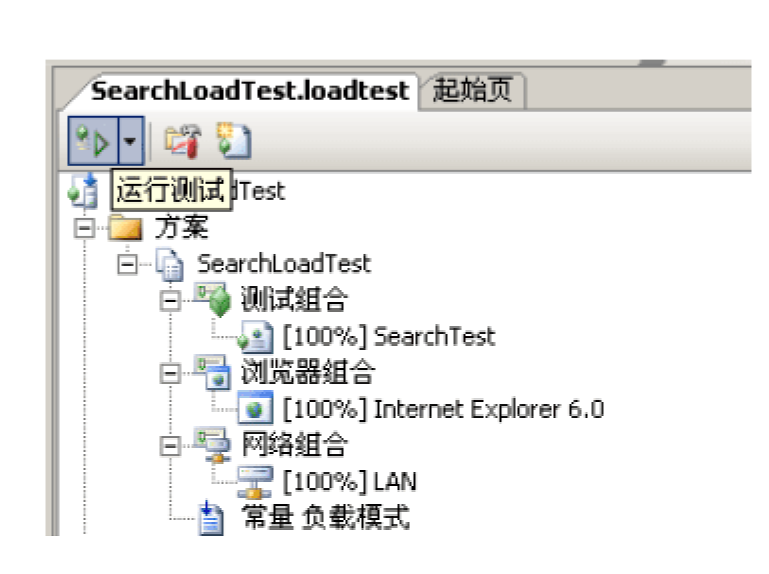

图 11-33　运行负载测试

3．分析测试结果

经过 1 分钟左右的运行后，可以得到如图 11-34 所示的负载测试结果。测试结果主要有测试报告、计数器、摘要和关系图等 4 部分组成。

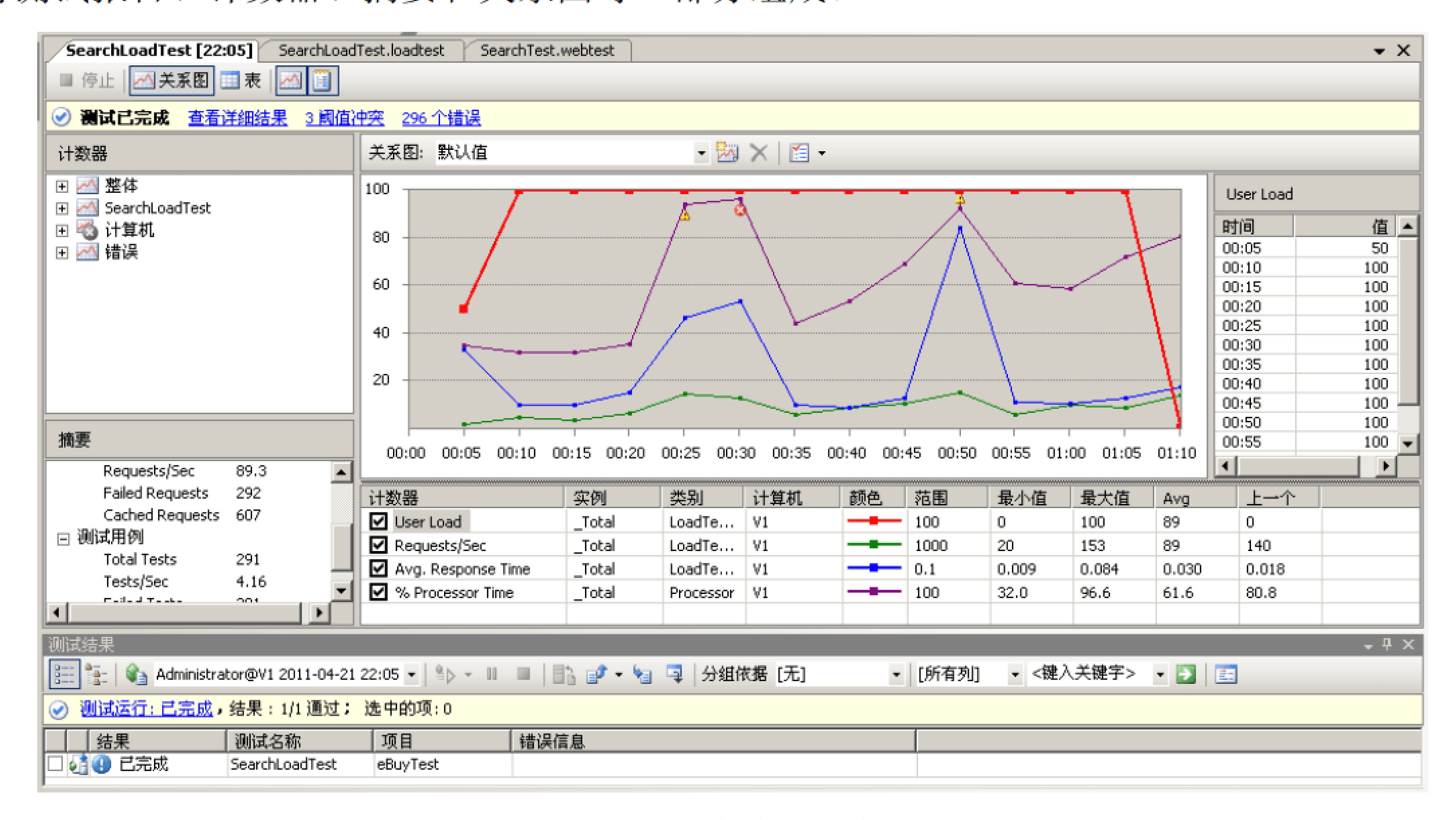

图 11-34　负载测试结果

图 11-34 中红色的折线（User Load）表示加载的用户数，在第 5 秒钟预热的时候已经开始加载 50 个用户，从第 10 秒钟到第 1 分零 5 秒加载的用户数量都为 100。绿色的折线（Requets/Sec）表示每秒钟发送的请求数。蓝色的折线（Avg.Response Time）表示在记录时间点的平均响应时间。而紫色的折线（% Processor Time）则表示占用处理时间的百分比。

这张关系图是分析系统的主要依据，从中可以看出系统在负载条件下的响应情况。

11.2.4 任务评价

标　　准	掌 握 情 况
能创建负载测试	熟练□ 基本□ 模糊□
能对负载测试进行合理的设置	熟练□ 基本□ 模糊□
能对负载测试结果进行基本的分析	熟练□ 基本□ 模糊□
自 我 小 结	

11.2.5 任务拓展

参照本节的内容，读者也可以对用户登录模块进行负载测试。通过设置不同的负载测试参数来获得不同的测试结果，并对结果进行分析。

本章小结

本章主要阐述了两种用于系统测试的不同测试方法：Web 测试和负载测试。Web 测试主要是通过测试 Web 系统的功能，来检验系统是否能够满足各项性能指标。在设计 Web 测试时，可以通过设置验证规则来达到不同的测试目的。负载测试则主要是通过模拟多用户的并发访问，来测试整个系统在某一硬件环境下的性能。同样，也可以通过各种测试参数的设置来模拟不同的测试条件，以达到不同的测试目的。

自我检测

一、选择题

1．下列关于 Web 测试的描述，错误的是（　）。

A．可以通过录制的方法来编辑 Web 测试

B．Web 测试只能用于测试某一个功能模块

C．Web 测试不能测试系统的性能

D．Web 测试可以用来测试系统的响应时间

2．下列关于负载测试的描述，错误的是（　）。

A．负载测试可以看作是性能测试的一种

B．负载测试只能对一个 Web 测试进行

C．负载测试可以模拟多个用户访问系统

D．负载测试可以设置测试的时间

3．下列关于负载测试结果分析的描述，错误的是（　）。

A．从负载测试的结果可以看出各记录时间点的请求数

B．从负载测试的结果可以看出各记录时间点的用户数

C．从负载测试的结果可以看出各记录时间点的平均响应时间

D．从负载测试的结果不能看出内存的消耗情况

4．下列 Web 测试用验证规则的说法，错误的是（　）。

A．在验证规则中可以设置最大响应时间

B．验证规则中的查找文本是用来验证响应中是否存在指定文本。

C．验证规则中的必需属性值是用来验证是否具有名称的 HTML。

D．验证规则中的窗体字段是用来验证是否存在具有指定名称和值的窗体字段。

5．下列关于测试的描述，正确的是（　）。

A．测试并不是一个重要的环节

B．测试都在系统开发完成后进行

C．测试就是指将系统运行后检查有没有错误

D．测试可以由非开发人员来执行

6．The following statements on Web test, which is wrong?（　）

A．We can record the operation to create a Web test

B．Web test can be only used for test one individual module

C．Web test cannot be used for test the performance of system

D．Web test can be used for test the response time of system

二、简答题

1．简述 Web 测试和负载测试的作用。

2．简述创建负载测试的一般步骤。该如何对负载测试进行设置？

参 考 文 献

[1] 张领．ASP.NET 项目开发全程实录[M]．北京：清华大学出版社，2008．
[2] 周新会．ASP 通用模块及典型系统开发实例导航[M]．北京：人民邮电出版社，2006．
[3] 张英男．ASP.NET 2.0 网络编程[M]．北京：电子工业出版社，2008．
[4] 孙玉宝．ASP.NET 程序设计实用教程[M]．北京：中国电力出版社，2008．
[5] 郝刚．ASP.NET 2.0 开发指南[M]．北京：人民邮电出版社，2006．
[6] 王有礼．ASP.NET 2.0 完全开发指南——基于 C#[M]．北京：科学出版社，2008．
[7] 陈承欢．Web 应用程序设计案例教程[M]．北京：清华大学出版社，2008．
[8] 陈平．软件设计师教程[M]．第 2 版．北京：清华大学出版社，2006．